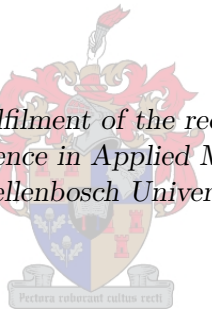
Adaptive occupancy grid mapping with measurement and pose uncertainty

by

Daniek Joubert



Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Applied Mathematics
at Stellenbosch University

Supervisors:

Prof. B.M. Herbst
Dr W.H. Brink

Applied Mathematics
Department of Mathematical Sciences
Faculty of Science

December 2012

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted if for obtaining any qualification.

Date: December 2012

Abstract

In this thesis we consider the problem of building a dense and consistent map of a mobile robot's environment that is updated as the robot moves. Such maps are vital for safe and collision-free navigation. Measurements obtained from a range sensor mounted on the robot provide information on the structure of the environment, but are typically corrupted by noise. These measurements are also relative to the robot's unknown pose (location and orientation) and, in order to combine them into a world-centric map, pose estimation is necessary at every time step. A SLAM system can be used for this task. However, since landmark measurements and robot motion are inherently noisy, the pose estimates are typically characterized by uncertainty. When building a map it is essential to deal with the uncertainties in range measurements and pose estimates in a principled manner to avoid overconfidence in the map.

A literature review of robotic mapping algorithms reveals that the occupancy grid mapping algorithm is well suited for our goal. This algorithm divides the area to be mapped into a regular lattice of cells (squares for 2D maps or cubes for 3D maps) and maintains an occupancy probability for each cell.

Although an inverse sensor model is often employed to incorporate measurement uncertainty into such a map, many authors merely state or depict their sensor models. We derive our model analytically and discuss ways to tailor it for sensor-specific uncertainty.

One of the shortcomings of the original occupancy grid algorithm is its inability to convey uncertainty in the robot's pose to the map. We address this problem by altering the occupancy grid update equation to include weighted samples from the pose uncertainty distribution (provided by the SLAM system).

The occupancy grid algorithm has been criticized for its high memory requirements. Techniques have been proposed to represent the map as a region tree, allowing cells to have different sizes depending on the information received for them. Such an approach necessitates a set of rules for determining when a cell should be split (for higher resolution in a local region) and when groups of cells should be merged (for lower resolution). We identify some inconsistencies that can arise from existing rules, and adapt those rules so that such errors are avoided.

We test our proposed adaptive occupancy grid algorithm, that incorporates both measurement and pose uncertainty, on simulated and real-world data. The results indicate that these uncertainties are included effectively, to provide a more informative map, without a loss in accuracy. Furthermore, our adaptive maps need far fewer cells than their regular counterparts, and our new set of rules for deciding when to split or merge cells significantly improves the ability of the adaptive grid map to mimic its regular counterpart.

Opsomming

In hierdie tesis beskou ons die probleem om 'n digte en konsekwente kaart van 'n mobiele robot se omgewing te bou, wat opgedateer word soos die robot beweeg. Sulke kaarte is van kardinale belang vir veilige, botsingvrye navigasie. Metings verkry vanaf 'n sensor wat op die robot gemonteer is, verskaf inligting rakende die struktuur van die omgewing, maar word tipies deur ruis vervorm. Hierdie metings is ook relatief tot die robot se onbekende postuur (posisie en oriëntasie) en, om hulle saam te voeg in 'n wêreldsentriese kaart, is postuurafskatting nodig op elke tydstap. 'n SLAM stelsel kan vir hierdie doeleinde gebruik word. Aangesien landmerkmetings en die beweging van die robot inherent ruiserig is, word die postuurskattings gekarakteriseer deur onsekerheid. Met die bou van 'n kaart moet hierdie onsekerhede in afstandmetings en postuurskattings op 'n beginselvaste manier hanteer word om te verhoed dat te veel vertroue in die kaart geplaas word.

'n Literatuurstudie van karteringsalgoritmes openbaar die besettingsroosteralgoritme as geskik vir ons doel. Die algoritme verdeel die gebied wat gekarteer moet word in 'n reëlmatige rooster van selle (vierkante vir 2D kaarte of kubusse vir 3D kaarte) en onderhou 'n besettingswaarskynlikheid vir elke sel.

Alhoewel 'n inverse sensormodel tipies gebruik word om metingsonsekerheid in so 'n kaart te inkorporeer, noem of wys baie outeurs slegs hulle model. Ons herlei ons model analities en beskryf maniere om sensorspesifieke metingsonsekerheid daarby in te sluit.

Een van die tekortkominge van die besettingsroosteralgoritme is sy onvermoë om onsekerheid in die postuur van die robot na die kaart oor te dra. Ons spreek hierdie probleem aan deur die opdateringsvergelyking van die oorspronklike besettingsroosteralgoritme aan te pas, om geweegde monsters van die postuuronsekerheidsverdeling (verskaf deur die SLAM stelsel) in te sluit.

Die besettingsroosteralgoritme word soms gekritiseer vir sy hoë verbruik van geheue. Tegnieke is voorgestel om die kaart as 'n gebiedsboom voor te stel, wat selle toelaat om verskillende groottes te hê, afhangende van die inligting wat vir hulle verkry is. So 'n benadering noodsaak 'n stel reëls wat spesifiseer wanneer 'n sel verdeel (vir 'n hoër resolusie in 'n plaaslike gebied) en wanneer 'n groep selle saamgevoeg (vir 'n laer resolusie) word. Ons identifiseer teenstrydighede wat kan voorkom as die huidige reëls gevolg word, en pas hierdie reëls aan sodat sulke foute vermy word.

Ons toets ons voorgestelde aanpasbare besettingsroosteralgoritme, wat beide metings- en postuuronsekerheid insluit, op gesimuleerde en werklike data. Die resultate dui daarop dat hierdie onsekerhede op 'n effektiewe wyse na die kaart oorgedra word sonder om akkuraatheid prys te gee. Wat meer is, ons aanpasbare kaarte benodig heelwat minder selle as hul reëlmatige eweknieë. Ons nuwe stel reëls om te besluit wanneer selle verdeel of saamgevoeg word, veroorsaak ook 'n merkwaardige verbetering in die vermoë van die aanpasbare roosterkaart om sy reëlmatige eweknie na te boots.

Acknowledgements

The author thanks the Harry Crossley Foundation as well as TATA Africa for funding this work.

Contents

Chapter 1

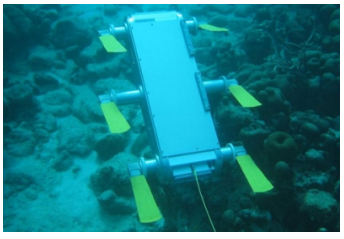
Introduction

Recent advances in technology, computer science and engineering have enabled the robots of today to conduct certain tasks without human guidance. As a result the idea of building a fully autonomous mobile robot has moved from science fiction to plausible reality.

Autonomous robots hold several advantages for mankind. Robots equipped with appropriate sensors and navigational software can explore environments that are unsafe or inaccessible for humans. Examples include the Mars exploration rovers [45; 64], search-and-rescue robots [53; 71] and mining robots [72]. Autonomous robots are also employed to assist the disabled and elderly [5].

A wide variety of autonomous robots exist. Figure 1.1 shows an example of an unmanned aerial vehicle (UAV), an underwater robot and a terrestrial vehicle.

(a) UAV



(b) underwater robot

(c) terrestrial vehicle

Figure 1.1 – Examples of a UAV (image from www.engagerc.com), an underwater robot (image from www.engadget.com) and a terrestrial vehicle (image from www.arrickrobotics.com).

1.1 Robotic localization and mapping

For a robot to be fully autonomous it has to function without assistance and, in particular, it has to be able to navigate safely through a possibly complex environment.

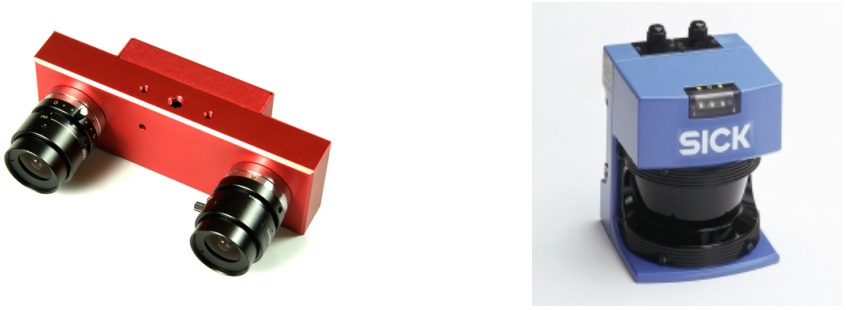
If the robot is able to perceive its environment in some way, a map can be constructed that can be interpreted by an obstacle detection and collision avoidance algorithm. If a map is available, path planning also becomes possible. Although exploration vehicles may be able to navigate their surroundings without a complete map, building one as the vehicle progresses is essential for the navigation process. Maps can also aid robots to navigate the same environment at a later stage. The focus of this thesis is on building maps for these purposes.

In order to build a map of the environment, the robot must have the ability to gather information about the structure of its surroundings. For this reason robots are usually equipped with sensors that enable them to perceive the environment and locate obstacles. Since the information gathered by a robot's sensors are relative to the robot, and not a global coordinate system, the robot needs to localize itself

within the map. On the other hand, the construction of a consistent map requires knowledge of the robot's location. This interdependency between localization and mapping poses a challenging problem.

Localizing the robot is typically performed by solving the simultaneous localization and mapping (SLAM) problem. A SLAM system identifies landmarks in the environment by means of some sensor (such as stereo cameras) and estimates both the landmark locations and the robot's position and orientation, relative to a fixed global coordinate system, by finding correspondences between landmarks over time. We refer to an estimate of the position and orientation of the robot as a pose estimate, and it covers the localization part of the problem. The map obtained from a SLAM system is typically quite sparse since it contains only salient landmarks. Such a map is therefore of little use for navigation, obstacle avoidance and path planning algorithms.

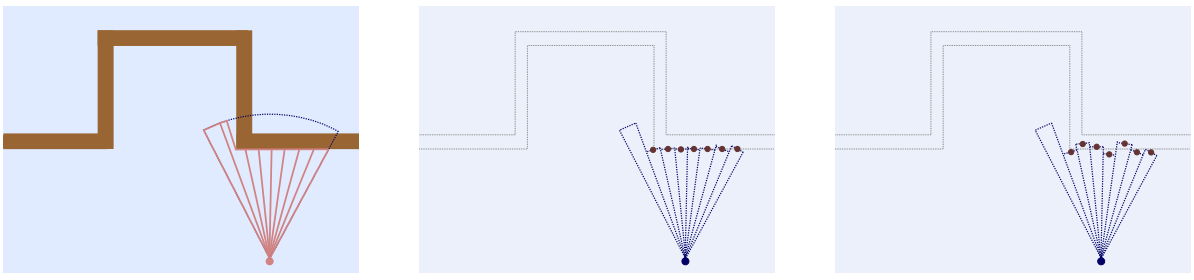
In order to build a dense map, the robot can be equipped with a range sensor that returns a set of distance measurements to obstacles in its field of view. Such a sensor is generally not used for SLAM due to the complexity of the correspondence problem. However, range sensors provide abundant information about the structure of the environment. Typical range sensors include stereo cameras and laser range finders, examples of which are shown in Figure 1.2.



(a) stereo cameras (b) laser range finder

Figure 1.2 – Examples of two sensors typically employed in robotic mapping. In (a) a stereo camera system is shown (image from http://paloma.isr.uc.pt) and (b) shows a 2D laser range scanner (image from www.sick.com).

Although some range sensors, such as sonar, return a single measurement to the first detected object within a beam, many have multiple beams and are therefore able to detect multiple objects. Consider the scenario depicted in Figure 1.3.



(a) environment (b) noise-free measurement (c) noisy measurement

Figure 1.3 – An environment is shown in (a). Obstacles are indicated in brown while the sensor and its field of view are shown in pink. A noise-free sensor will return a configuration of point obstacles as shown in (b). If measurement noise is present these points may have a configuration as shown in (c).

Figure 1.3(a) shows an environment with obstacles, as well as a sensor and its field of view which, in this case, consists of multiple beams. If the sensor is noise-free, the measurement taken results in a configuration of point obstacles as shown in (b). In reality, however, measurement noise may be present. An example of an obstacle configuration arising from the measurement corrupted by random noise is shown in (c). This means that, if we want to utilize measurements captured by a range sensor in order to build a map, we should model and incorporate measurement uncertainty.

Since range measurements are given relative to the robot, we require a pose estimate every time a measurement is to be incorporated into the map. Robot motion is characterized by noise and uncertainty since the instructions given to the robot's actuators may not be carried out perfectly. Therefore, when building the map, we should also attempt to incorporate the uncertainty associated with pose estimates. If this uncertainty is not handled in a principled manner, the robot may become overconfident which can lead to collisions.

We proceed with an overview of published research on the robotic mapping problem.

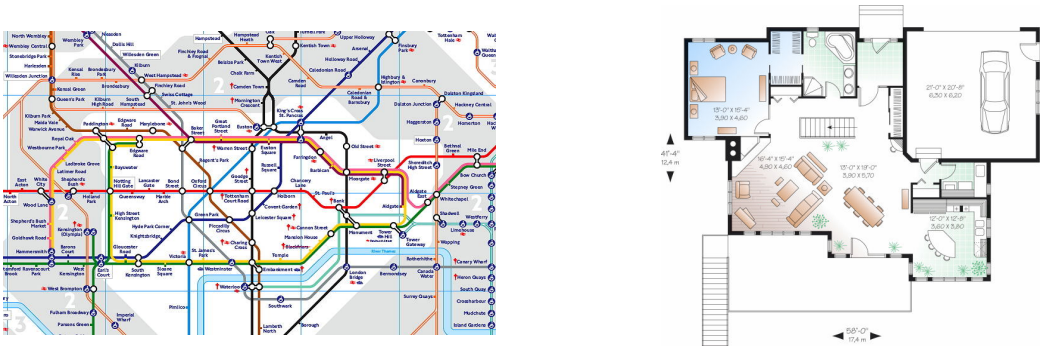
1.2 Overview of robotic mapping

The problem of robotic mapping has been an active research area since the 1980s and yet many open research problems remain. Mapping large environments, handling robot uncertainties as well as mapping dynamic environments are but a few of these problems. In this section we provide a brief overview of approaches and algorithms from the literature.

1.2.1 Map representation

The first point to consider when designing a robotic mapping algorithm is to find a map representation best suited for the application. Maps obtained from robotic mapping algorithms can be divided roughly into two categories, namely metric and topological maps.

Topological maps are simplified graph-like representations of the environment in which significant places or landmarks are vertices and the edges connecting them correspond to safe, obstacle-free pathways. Although many approaches for generating topological maps exist [39; 58; 60] the maps are typically not as detailed as metric maps and uncertainty representation can be problematic. These types of maps also struggle to differentiate between distinct places [97]. A well-known example of a topological map is the London Underground map shown in Figure 1.4(a).



(a) topological map (b) metric map

Figure 1.4 – The London Underground map in (a) is an example of a topological map (image from www.tfl.gov.uk), while the blueprints of a house, shown in (b), is an example of a metric map (image from www.freeblueprints.net).

Metric maps, on the other hand, provide specific, detailed geometric information about the environment and are therefore more useful to autonomous robots. As a result the metric map has become a popular

way of representing a robot's environment [17; 23; 49; 94; 107]. An example of a metric map is shown in Figure 1.4(b). Hybrid maps combining the best of both representations have also been proposed [91; 97].

Apart from the distinction between metric and topological maps, maps can also be classified as world-centric or robot-centric. World-centric maps are given relative to some global reference frame, while robot-centric ones are given relative to the robot's current position. Robot-centric maps are easier to build in the sense that no coordinate transformation, and therefore no relative movement information from one time step to the next, is necessary. However, these maps tend to struggle to distinguish between parts of the environment that are similar [93] and are of little use after the robot has left that particular area. For these reasons we will focus on world-centric maps.

1.2.2 Localization

We mentioned that there exist interdependencies between the mapping and localization processes. SLAM attempts to solve this problem probabilistically, and has been studied extensively [4; 33; 46; 61; 67]. Such a system may, however, suffer from computational complexity and it remains a challenge to build a complete map of the environment.

In pure mapping algorithms the pose information is usually obtained from some other system (such as SLAM). Many algorithms exist that calculate a pose estimate without providing information regarding uncertainty. The iterative closest point (ICP) algorithm [10], for example, determines the rotation and translation required to align two surfaces, obtained from two consecutive range measurements, that partially overlap.

In our case, however, we prefer to incorporate pose uncertainty into the mapping algorithm, and therefore some measure of the uncertainty associated with a pose estimate is essential. Since the SLAM problem is formulated probabilistically, our aim is to use the localization output from such a system as an input to our mapping algorithm.

1.2.3 Robotic mapping algorithms

In [93] five types of algorithms that dominate the robotic mapping domain for static environments are discussed and compared, namely Kalman filter approaches, expectation maximization (EM) algorithms, hybrid approaches of these two, object maps and occupancy grid maps. In this section we present some key aspects of each of these algorithms. For a more comprehensive survey of these methods the reader is referred to [93].

For a map to be useful it should be amenable to obstacle avoidance, path planning and general navigation algorithms. As a result most state-of-the-art mapping algorithms are probabilistic [93], a property that aids the incorporation of pose and measurement uncertainties.

Kalman filter approaches [25] attempt to solve the SLAM problem by employing Bayes filters and representing the posterior probabilities as Gaussian distributions. Maps resulting from such approaches are generally point-based and incomplete.

EM algorithms function by maximizing the posterior through a hill-climbing algorithm [95]. For convergence these algorithms typically require multiple iterations through the data and cannot be applied incrementally (as the robot moves through the environment).

Hybrid approaches between the EM algorithm and the Kalman filter are known as incremental maximum likelihood algorithms [90]. These algorithms build a map incrementally as the measurements are received but do not keep track of uncertainty, akin to an EM algorithm with an M-step but not an E-step, and in some cases fail completely [93].

In object mapping algorithms the environment is represented by sets of line segments which implies that if the environment is too intricate to be represented by lines, they fail to generate a complete map. An extension of this idea is proposed in [26] where the geometry of the environment is represented by polyhedral sets. Some approaches build triangle mesh-based maps [99; 101; 55] but they have difficulties in representing uncertainty.

The author of [29] introduces the notion of representing a single range image as an octree where each node is labelled as unseen, empty or occupied. This idea paved the way for a pioneering algorithm

introduced by Moravec and Elfes in [69], which is known as the occupancy grid algorithm. In this algorithm the area to be mapped is divided into a regular grid of cells and each is assigned an occupancy probability. According to [93] the occupancy grid algorithm provides distinct advantages over other static environment algorithms. The main shortcoming of the algorithm is that pose uncertainty is not incorporated into the map since the map is built by assuming that the most likely pose at a particular time step is correct [96].

A method similar to the occupancy grid algorithm has been proposed that extracts an isosurface from a 3D grid of cells. This is done by assigning a value to each cell, that indicates the signed distance to the nearest obstacle, and extracting the zero crossing as the final surface [31; 54; 103]. Incorporating uncertainty into this method is problematic due to the fact that a hard assignment is made (a cell is either part of the isosurface or it is not).

After assessing the advantages and shortcomings of each of these algorithms, we come to the conclusion that the occupancy grid algorithm will form the basis on which we build our mapping algorithm.

1.2.4 Motivation for the occupancy grid algorithm

Since robotic mapping is characterized by noise in the pose estimates as well as the measurements, it is vital to incorporate uncertainties into the mapping algorithm. Occupancy grids, which are probabilistic in nature, have been widely employed in robotic mapping applications where a complete map of the environment is required [18; 34; 41; 47; 109]. The algorithm has also been used to build consistent maps using inputs from multiple robots [11; 24].

Many authors have investigated the use of occupancy grids for navigational purposes. In [17] the idea of virtual force fields is introduced, where obstacles are considered to have a repulsive effect on the robot. Fuzzy logic operators have also been included to represent sensor uncertainty [73]. Occupancy grids are also used for obstacle detection or avoidance [43; 56; 78], path planning [27] and frontier exploration [106; 98]. An extensive survey of algorithms for visual navigation, that also includes occupancy grids, is provided in [16].

There exists extensive literature on adapting the occupancy grid algorithm for specific needs. The authors of [48], for example, build 3D maps from segmented range data obtained with stereo vision. A 2D occupancy grid map forms the basis of a map-building algorithm proposed in [74] where measurement uncertainty is designed specifically for a sonar sensor, and readings are fused by using the Dempster-Shafer formulation rather than the Bayesian approach of the standard occupancy grid algorithm.

The occupancy grid algorithm is popular partly because it provides an explicit representation of free (or open) space, in the sense that grid cells can be labelled as free. It is also not limited to Gaussian sensor noise but can accommodate any noise distribution. Occupancy grids have also been used in localization applications by comparing consecutive robot-centric occupancy grid maps and estimating the transformations between them [6]. The authors of [82] provide a comparison of such localization techniques.

Another reason for the occupancy grid algorithm's popularity stems from the fact that the update equations are relatively easy to implement because new information can be incorporated into a current map by a simple addition of log odds ratios. Also, some mapping algorithms can converge to local minima which the occupancy grid algorithm manages to avoid [93]. Furthermore, it is an incremental algorithm which is essential if we want to update the map as the robot moves through the environment.

1.3 Problem statement

The problem we focus on in this thesis involves building a consistent map of a static environment. In order to achieve this we assume the robot is controlled, either by a human or an autonomous path planner, to move through an environment to be mapped.

As input to our system we will have, at every time step, an estimate of the robot's pose and the associated uncertainty that comes from a SLAM system. If the robot moves in 2D the pose consists of a three-dimensional vector $[\,x\ y\ \theta\,]^{\mathrm{T}}$ and in 3D it is given by $[\,x\ y\ z\ \theta\ \phi\ \psi\,]^{\mathrm{T}}$. Additionally, we will have measurements captured by a range sensor mounted on the robot. Such a measurement can be in the

form of a 1D signal coming from, for example, a laser range scanner with multiple beams corresponding to 2D rays, or a 2D signal in the form of a range image where the beams correspond to rays in 3D.

From these inputs we want to build a consistent map of the environment that is updated as the inputs are received. Also, since robotic mapping algorithms typically form part of an integrated system we should keep in mind that the map must be useful for obstacle avoidance, path planning and other navigational purposes. We choose to focus on the occupancy grid algorithm as it is ideally suited for this task.

Since measurement noise can be present in range sensor readings, and the noise is typically not negligible, we want to incorporate measurement uncertainty into the map building process. To our knowledge, there exists no generic formulation of incorporating sensor-specific measurement noise into an occupancy grid. We pay specific attention to the development of a generic sensor model for the occupancy grid algorithm that is capable of modelling measurement uncertainty for a variety of range sensors.

We mentioned that the main shortcoming of the basic occupancy grid algorithm is its inability to handle pose uncertainty. We want to address this shortcoming because pose uncertainty from a SLAM system is also generally not negligible and may have an influence on how the environment perceived by the sensors is translated to the map. Of course, we have to assume that the modelling of pose uncertainty by the SLAM system is realistic.

The occupancy grid algorithm has sometimes been criticized for its memory usage [16], since numerous cells and their probabilities have to be stored, and the algorithm struggles to map large environments effectively. We will consider a way of representing the map as a quadtree (in 2D) or octree (in 3D) to allow cells in the map to have varying sizes. These representations can enable the use of fewer cells, ideally without a significant loss in detail, and thus decrease memory requirements and computation.

1.4 Outline of our approach

We begin by providing a brief background on the SLAM problem, and some algorithms that are popularly employed, in Chapter 2. This is important for the design of our mapping algorithm that will use the localization output from the SLAM system as pose estimates, and assumptions made by the SLAM algorithm are therefore also applicable to our system.

In Chapter 3 we introduce the basic occupancy grid algorithm, provide a derivation and discuss an important independence assumption made to reduce the dimensionality of the mapping problem. The algorithm presented in that chapter forms the basis on which our work is built.

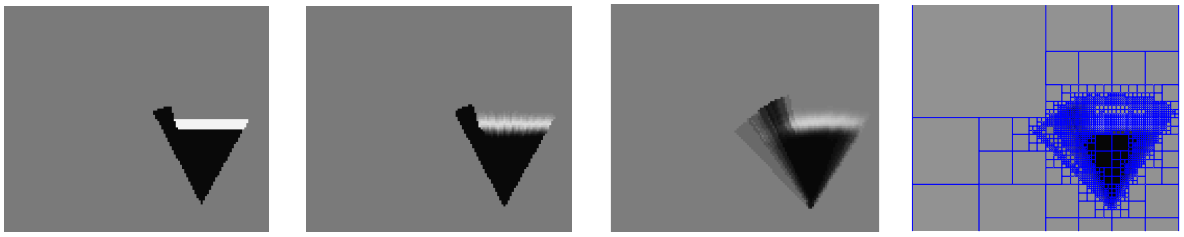
We assume that the robot is equipped with a range sensor that is able to perceive the environment and return information about the location of obstacles relative to the robot. These measurements may be corrupted by noise, and in Chapter 4 we discuss a means of modelling and incorporating the uncertainty associated with such measurements into the basic occupancy grid algorithm.

Besides measurement uncertainty, pose uncertainty is something that must be dealt with in robotic localization and mapping. Currently, the occupancy grid algorithm does not incorporate pose uncertainty and in Chapter 5 we address this problem by deriving a new update equation that makes use of samples generated from a pose uncertainty distribution.

The issue of memory efficiency is addressed in Chapter 6 where we extend the occupancy grid algorithm to accommodate varying grid sizes. The idea is based on work by Einhorn et al. [37] and we make some improvements by eliminating certain inconsistencies that can arise.

Figure 1.5 provides a preview of the key concepts introduced in Chapters 3 to 6. For this example the same environment and field of view as in Figure 1.3(a) are used to simulate a measurement at a single time step. In (a) the map resulting from the basic occupancy grid algorithm is shown, where noise is assumed to be absent in both the measurement and the pose. Here the lighter the colour of the cell, the higher its occupancy probability. If measurement noise is incorporated an occupancy grid map such as the one shown in (b) is obtained. After incorporating both measurement and pose uncertainty the resulting occupancy grid map may be similar to the one shown in (c). Finally, by adapting the grid size, the occupancy grid map has multiple resolutions such as the one shown in (d).

We test our proposed algorithm on various simulated datasets and, according to the results, our algorithm improves on the basic occupancy grid algorithm used for robotic mapping. From a comparison between



(a) basic algorithm (b) measurement unc. (c) pose uncertainty (d) adaptive grid

Figure 1.5 – A preview of the key concepts introduced in Chapters 3 to 6, which range from the basic occupancy grid algorithm, the incorporation of measurement uncertainty, the incorporation of pose uncertainty and adapting the grid size based on the information to be stored. In these 2D maps black indicates free space and white regions are occupied by obstacles in the environment. This example corresponds to the situation depicted in Figure 1.3(a).

the algorithm presented in [37] and our improvements we find that our algorithm performs significantly better. We also test our algorithm on a real-world dataset and obtain promising results. Details of these tests and results are presented in Chapter 7, which is followed by concluding remarks in Chapter 8.

Chapter 2

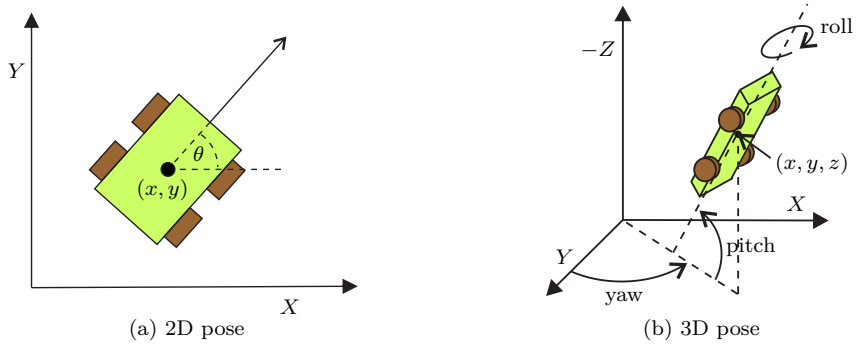
Simultaneous localization and mapping

In order to build a map we require a sensor, typically mounted on the robot, to gather information about the structure of the environment. As the robot moves through an area to be mapped, the sensor provides measurements of the environment relative to the robot. In order to use this information, it is vital to know where the robot is relative to some global coordinate system when a particular measurement is taken. On the other hand, for the robot to localize itself within that global coordinate system, a map of the environment is required. This problem of interdependency has received much attention in the literature [4; 33; 46; 61; 67] and is commonly known as the simultaneous localization and mapping (SLAM) problem.

This chapter provides some background on the SLAM problem. We will use the localization output from a SLAM system as input to our grid mapping algorithm, which we handle completely separately. It is therefore important to understand the general ideas behind the algorithms commonly used to solve the SLAM problem as well as the format of the localization output. We also need to be aware of assumptions made by these algorithms that are also relevant to our mapping algorithm.

2.1 The SLAM problem

In order to map the environment using the given measurements, we have to localize the robot and in order to do that a map is required. This localization is often termed pose estimation. The pose of a robot moving on a plane (in 2D) is described by $[\,x\ y\ \theta\,]^{\mathrm{T}}$, where x and y represent the position of the robot in some fixed global coordinate system and θ is the angle between the bearing of the robot and the positive x-axis, as shown in Figure 2.1(a). The pose of a robot moving in 3D is given by a three-element position vector (x, y, z) as well as three angles (roll, pitch and yaw), and is illustrated in Figure 2.1(b).



(a) 2D pose (b) 3D pose

Figure 2.1 – The parameters describing a robot's pose in 2D and 3D.

The map estimated by a SLAM system is generally in the form of position coordinates of features or landmarks observed by a sensor mounted on the robot. Commonly used sensors include laser range

scanners, sonar and vision-based systems such as stereo cameras [3]. In the SLAM framework maps are usually sparsely populated to reduce computation [3; 96]. For this reason, dense mapping of the environment is typically not combined with the SLAM problem but performed separately, as in our approach.

In general, pose estimation requires a solution to the correspondence problem. If a sensor observes some features and the system is able to verify that they have already been seen, the relative movement of the robot from the previous time step to the current one can be determined. When stereo cameras are employed, an algorithm such as SIFT [62] or SURF [8] can contribute to solving the correspondence problem.

The map and pose estimations are usually corrupted by noise originating from imperfections embedded in the mechanism of the sensor, environmental factors, incorrect correspondences or the fact that instructions given to the robot's actuators may not be executed perfectly. Incorporating uncertainty into the SLAM algorithms is therefore advisable, and this also applies to our mapping strategy.

Since modelling the uncertainty in the system is important, it is natural to formulate the SLAM problem using probabilities. There are generally two forms of the SLAM problem: one is called the online SLAM problem and the other the full SLAM problem. The full SLAM problem asks for the posterior probability of all poses along with the entire map, i.e.

$$p(x_{1:t}, m|y_{1:t}, u_{1:t}), \tag{2.1.1}$$

where $x_{1:t}$ denotes the pose estimates from the first time instance up to time t, m the map, $y_{1:t}$ the sensor measurements and $u_{1:t}$ the control instructions sent to the robot (in an attempt to control its pose). A custom of the SLAM community is to use the letter z for denoting measurements but, since the focus of this thesis is on dense mapping, we follow the standard notation of the occupancy grid algorithm. We therefore reserve z for range measurements (which differ from the feature-based measurements used by the SLAM system). Since the full SLAM problem is formulated in such a way that it seeks an estimate over all the previous measurements and controls, it is generally not solved incrementally and is therefore not applicable to our problem.

The online SLAM problem requires the posterior over the current pose x_t along with the map, and is formulated as

$$p\left(x_t, m|x_{1:t-1}, u_{1:t}, y_{1:t}\right). \tag{2.1.2}$$

Algorithms that solve this problem function under a Markov assumption, which postulates that the current state is dependent only on the previous state and the current measurement and control, so that (2.1.2) becomes

$$p(x_t, m|x_{t-1}, u_t, y_t). \tag{2.1.3}$$

Solving the online SLAM problem therefore requires an incremental algorithm that discards previous measurements and controls after they have been processed [96]. For this reason we focus on algorithms that solve the online SLAM problem.

Because of the continuous nature of the pose estimation variables it is intractable to calculate the full posterior in (2.1.3). In practice, SLAM algorithms rely on approximations. We proceed to discuss some popular SLAM algorithms and the assumptions associated with them.

2.2 EKF SLAM

It was mentioned that the measurement as well as the control update process can be corrupted by noise. An assumption made when employing the extended Kalman filter (EKF) to solve the SLAM problem is that Gaussian noise models are applicable in both processes [96].

2.2.1 Algorithm description

The input to the EKF SLAM algorithm at time t is the previous estimate of the state vector μ_{t-1}, an associated covariance matrix Σ_{t-1}, the current control input u_t and the current measurement y_t. The state vector contains all variables that are estimated by the EKF. In this case it consists of the pose information as well as the locations of the landmarks in the map, and so has dimension $3 + 2N$ for a

robot moving in 2D ($6+3N$ in 3D) where N is the number of landmarks. The output of the algorithm is a new mean state vector μ_t with an associated covariance matrix Σ_t. This mean and covariance describe a $(3+2N)$-dimensional Gaussian distribution.

Since the control input causes a change in the pose, it results in a change from the previous state vector μ_{t-1} to the current one μ_t. We assume that there exists a function g so that

$$\mu_t = g(u_t, \mu_{t-1}) + \epsilon_t, \tag{2.2.1}$$

where ϵ_t denotes the noise associated with the control process. The measurement provides additional information about both the pose and the landmarks contained in the state vector, and we may assume that a function h exists that relates the new state vector and the measurement as

$$y_t = h(\mu_t) + \delta_t, \tag{2.2.2}$$

where δ_t denotes the noise associated with the measurement process. If the functions g and h are linear in their arguments, a normal Kalman filter can be employed.

The EKF SLAM algorithm starts by predicting a new state vector, denoted by $\hat{\mu}_t$, by incorporating the control according to

$$\hat{\mu}_t = g(u_t, \mu_{t-1}). \tag{2.2.3}$$

The functions g and h may very well be nonlinear and, since the Kalman filter is not equipped for that, the EKF proceeds to linearize these functions with a first-order Taylor approximation. We introduce a Jacobian matrix G_t that contains the partial derivatives of the components of g with respect to the variables in u_t and μ_{t-1}. We also determine a Jacobian matrix H_t from the partial derivatives of h with respect to $\hat{\mu}_t$.

The predicted covariance $\hat{\Sigma}_t$ associated with $\hat{\mu}_t$ is then calculated as

$$\hat{\Sigma}_t = G_t \Sigma_{t-1} G_t^{\mathrm{T}} + R_t, \tag{2.2.4}$$

where R_t is the covariance matrix associated with the Gaussian noise in the control update process. The Kalman gain, which is an indication of the extent to which the measurement is incorporated into the final estimation, is calculated as

$$K_t = \hat{\Sigma}_t H_t^{\mathrm{T}} \left(H_t \hat{\Sigma}_t H_t^{\mathrm{T}} + Q_t \right)^{-1}, \tag{2.2.5}$$

where Q_t is the covariance matrix associated with the Gaussian measurement noise. The predicted state vector is now updated by incorporating the measurement y_t according to

$$\mu_t = \hat{\mu}_t + K_t \left(y_t - h(\hat{\mu}_t) \right). \tag{2.2.6}$$

The associated covariance matrix at time t is given by

$$\Sigma_t = (I - K_t H_t) \hat{\Sigma}_t, \tag{2.2.7}$$

where I is the identity matrix. For more detail on this algorithm, and for a derivation of the update equations stated here, the reader is referred to [96]. Different variants of the algorithm exist, depending for example on whether or not the landmark correspondence problem is solved in advance, and details can also be found in [96].

2.2.2 Discussion

Figure 2.2 offers a crude representation of EKF SLAM output at one time step. The joint posterior of the robot pose and landmark locations is Gaussian, so it is natural to think of the uncertainties in these states as confidence ellipses in the 2D plane (or ellipsoids in the case of 3D SLAM). These uncertainties are in fact given in the form of a single high-dimensional Gaussian distribution but, for visualization purposes, it is convenient to depict marginalized distributions of the robot pose and every landmark location separately (as we do in the figure). Note also that, since we plot these ellipses in 2D, the uncertainty in the robot's orientation θ is not shown explicitly.

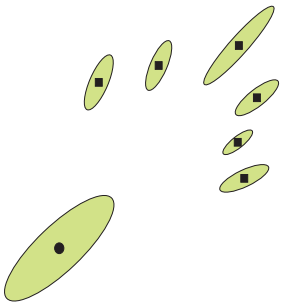


Figure 2.2 – Representation of uncertainty at one time step of EKF SLAM. Mean landmark locations are indicated by the black squares and the mean robot position by the black circle. The ellipses are indications of uncertainty.

Apart from the Gaussian noise assumption, there are some other limitations in the EKF SLAM algorithm that should be noted. Because of its inability to handle multi-modal distributions to any degree of accuracy, the algorithm tends to perform poorly in the presence of ambiguous landmarks without clear, distinct characteristics [96]. The algorithm may fail completely if the noise is far from Gaussian or if the dynamics of the system are highly nonlinear. Another point to note is that negative information cannot be processed, i.e. if a landmark is not observed, it does not add any information to other landmark locations or to the pose estimation. If a landmark is seen in multiple views, however, the estimate of its location will improve.

Despite its limitations, EKF SLAM is relatively simple to implement and has been widely used in efforts to solve the SLAM problem [28; 83; 84].

2.3 The FastSLAM algorithms

In EKF SLAM the noise is assumed to be Gaussian, and the control and measurement equations are linearized in order to use the Kalman filter, which results in a Gaussian posterior distribution. In general, this assumption may be invalid. The FastSLAM algorithms employ a particle filter, which does not assume a particular distribution but rather represents the posterior by a set of weighted samples (called particles). For SLAM in particular, the Rao-Blackwellized particle filter is utilized. This filter uses particles to represent the posterior over some variables along with Gaussians (or some other distributions) to represent the remaining variables [96].

In FastSLAM this split in variables occurs between the pose variables and the variables that represent the landmark locations. A particle filter is used to estimate the robot's pose while the landmark locations are estimated individually, and for each particle, by separate low-dimensional EKFs. This is possible under the assumption that the landmarks are conditionally independent, given the robot's pose, and can therefore be handled separately [96].

2.3.1 Algorithm description

Each particle in the filter corresponds to an estimated pose, and we will denote the kth particle by $x_t^{[k]}$. Since a particular pose results in a particular configuration of landmark locations, each particle is also accompanied by a set of N EKFs, where N is the number of landmarks, each with a mean vector $\mu_{j,t}^{[k]}$ and covariance matrix $\Sigma_{j,t}^{[k]}$ for the jth landmark.

The input to FastSLAM is a measurement y_t at time t, a control update u_t and a set of particles X_{t-1} that represent the previous posterior.

From each of the particles in X_{t-1}, a pose $x_{t-1}^{[k]}$ is retrieved. By assuming this pose was correct, a new pose $x_t^{[k]}$ is predicted from the control update by sampling from

$$p(x_t|x_{t-1}^{[k]}, u_t). \tag{2.3.1}$$

All landmarks observed at the current time step are matched to landmarks already in the map. If an observed landmark y_t^i corresponds to landmark j, the measurement y_t^i is incorporated into the estimated location of landmark j by updating the mean $\mu_{j,t}^{[k]}$ and the associated covariance $\Sigma_{j,t}^{[k]}$ through an EKF measurement update step. If the landmark is observed for the first time, a new EKF is initialized. Together with the new pose $x_t^{[k]}$, the particle's information is now complete, except for the fact that the likelihood of this particle representing the true state may have changed. This can be rectified by assigning an importance weight $w_t^{[k]}$ to the new particle based on the extent to which it supports the measurement [96], and is calculated according to

$$\begin{aligned} w_t^{[k]} &= \frac{p\left(x_{1:t}^{[k]} \middle| y_{1:t}, u_{1:t}\right)}{p\left(x_{1:t}^{[k]} \middle| y_{1:t-1}, u_{1:t}\right)} \\ &= \frac{p\left(y_t \middle| x_t^{[k]}\right)}{p\left(y_t \middle| y_{1:t-1}, u_{1:t}\right)}. \end{aligned} \tag{2.3.2}$$

Once all particles are propagated to the next time the particles are resampled with replacement. The reason for this step stems from the fact that the importance weights of many particles, when propagated, can become extremely small and therefore contribute virtually nothing to the representation of the final posterior. By resampling with replacement, only the particles with high importance weights (hence the more likely particles) survive.

In short, this is the FastSLAM 1.0 algorithm. FastSLAM 2.0 is essentially the same, with one exception. When sampling the new pose $x_t^{[k]}$, FastSLAM 2.0 incorporates the control update u_t as well as the measurement y_t. FastSLAM 1.0 is ineffective when the uncertainty associated with the control is large relative to the measurement uncertainty. Particles are then generated based only on the relatively inaccurate control, but only a small subset of those particles may agree with the measurement. Since the measurement is used to assign importance weights, very few particles will survive. FastSLAM 2.0 samples poses in conjunction with the measurement so that more particles survive at every time step. This means that, overall, fewer particles are required and it results in a more efficient algorithm.

As in the case of EKF SLAM, variants exist depending on whether or not the correspondence problem is solved beforehand. Further details on the implementation and general working of the FastSLAM algorithms can be found in [96].

2.3.2 Discussion

Figure 2.3 gives a rough depiction of the FastSLAM output at one time step. Here, contrary to EKF SLAM where the robot's pose is described by a Gaussian distribution, we plot the particles at their locations (again, robot orientation is not shown) and colour them according to their importance weights. This yields some indication of the spread of the robot pose posterior. Note that in the figure this distribution appears to be multi-modal, which a single Gaussian cannot represent. The landmark locations are described by Gaussian distributions so we can draw confidence ellipses for them. Since each particle has a different configuration of landmark locations, we plot only the configuration associated with the most likely particle for clarity.

Besides the fact that the FastSLAM algorithms are not restricted to a specific noise probability distribution, they are also able to cope with nonlinear models [3] whereas the EKF approximates these models through linearization. The result is a key advantage in cases where the pose uncertainty is relatively high or where the kinematics of the system are highly nonlinear [96]. FastSLAM can also solve both the full and the online SLAM problems, and to date is the only algorithm able to do so. On the other hand, the algorithm may continue to be inefficient, display significant growth in complexity with regards to the number of particles used, and can become unstable in large environments [3]. Despite their complexities, the FastSLAM algorithms have also been widely employed to solve the SLAM problem [7; 81; 86].

The EKF SLAM and FastSLAM algorithms are two of many that exist to solve the SLAM problem. We highlight these two specifically because, respectively, they employ two fundamental ways of representing the uncertainty in state estimates: parameters of a closed-form probability distribution function (a Gaussian distribution in the case of EKF SLAM) and a sample-based representation (weighted particles

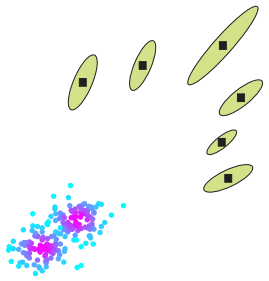


Figure 2.3 – Representation of uncertainty at one time step of FastSLAM. As before, the uncertainty in the location of the landmarks is indicated by ellipses. The dots represent the robot location estimates as given by the particles. They are coloured according to their weight (the warmer the color the larger the weight).

in the case of FastSLAM). The techniques we develop in the chapters that follow allow for either one of these representations. For a more detailed survey on various other SLAM algorithms, the reader is referred to [35; 36].

2.4 Input to a grid mapping algorithm

As mentioned, we are interested in the localization output of a SLAM system. For this reason we explicitly state the input to our system at time step t, which depends on the SLAM algorithm utilized for localization. We also have as input to our system the raw measurement obtained directly from the range sensor mounted on the robot, which is not necessarily the same sensor employed in the SLAM process.

2.4.1 Input from EKF SLAM

The output of EKF SLAM is a mean vector μ_t with associated covariance matrix Σ_t. The mean state vector consists of a localization and a mapping part and can therefore be partitioned as

$$\mu_t = \begin{bmatrix} \mu_{t,x} \\ \mu_{t,m} \end{bmatrix}, \tag{2.4.1}$$

where $\mu_{t,x}$ represents the localization part and $\mu_{t,m}$ the mapping part of the state vector at time t. Similarly, the covariance matrix can be partitioned into submatrices according to

$$\Sigma_t = \begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xm} \\ \Sigma_{t,mx} & \Sigma_{t,mm} \end{bmatrix}. \tag{2.4.2}$$

Note that since Σ_t is symmetric, $\Sigma_{t,xm} = \Sigma_{t,mx}$.

The mean vector μ_t and covariance matrix Σ_t describe the joint probability distribution

$$p\left(x_t, m \,|x_{t-1}, u_t, y_t\right), \tag{2.4.3}$$

where x_t represents the pose of the robot and m the map. In order to extract the pose information from this joint distribution we require the marginal distribution

$$p\left(x_t \,|x_{t-1}, u_t, y_t\right) = \int p\left(x_t, m \,|x_{t-1}, u_t, y_t\right) \, dm. \tag{2.4.4}$$

Since $p(x_t, m| \cdot)$ is Gaussian, it can be shown that $p(x_t| \cdot)$ is also Gaussian [12] and is therefore also completely described by a mean vector and a covariance matrix. With partitions such as those in (2.4.1) and (2.4.2), it can also be shown [12] that the mean corresponding to $p(x_t| \cdot)$ is equal to $\mu_{x,t}$ and the associated covariance is given simply by $\Sigma_{t,xx}$. The vector $\mu_{x,t}$ and covariance matrix $\Sigma_{t,xx}$ form the localization input to our system at time t.

2.4.2 Input from FastSLAM

With either of the FastSLAM algorithms the output is a set of particles with pose information as well as importance weights. Furthermore, each particle has a set of N mean vectors and covariance matrices describing the landmark locations but, since we are interested only in the localization, we discard this information.

Recall from section 2.3.1 that particles are resampled with replacement before continuing to the next time step. During this procedure particles are sampled with a probability proportional to their importance weights. This means that particles with small weights are most likely to be discarded and particles with large weights are duplicated. After resampling all particles have equal importance weights. The reason behind this step is to increase the likelihood that a high number of particles will survive the next time step. Also, by discarding particles with small weights, virtually no information is lost since they contribute little to the representation of the posterior distribution. Consequently we can utilize the particles before the resampling step as input to our algorithm, and discard those that would probably have been eliminated through the resampling step, or utilize the particles after resampling, whichever is most convenient.

We extract the pose information from each of the M particles and denote the collective set of poses by

$$x_t = \left\{ x_t^{[1]}, x_t^{[2]}, \cdots, x_t^{[M]} \right\}. \tag{2.4.5}$$

These particles also have accompanying importance weights

$$w_t = \left\{ w_t^{[1]}, w_t^{[2]}, \cdots, w_t^{[M]} \right\}, \tag{2.4.6}$$

which is an indication of the likelihood of each particle. The sets x_t and w_t form the localization input to our mapping algorithm at time t.

In this chapter a brief description of the SLAM problem and some algorithms commonly used in attempts to solve it was supplied. Now that we have explicitly stated the output from such a system, which is used as input to our grid mapping algorithm, we proceed to describe the occupancy grid mapping algorithm, which forms the basis for all subsequent work.

Chapter 3

Basic occupancy grid mapping

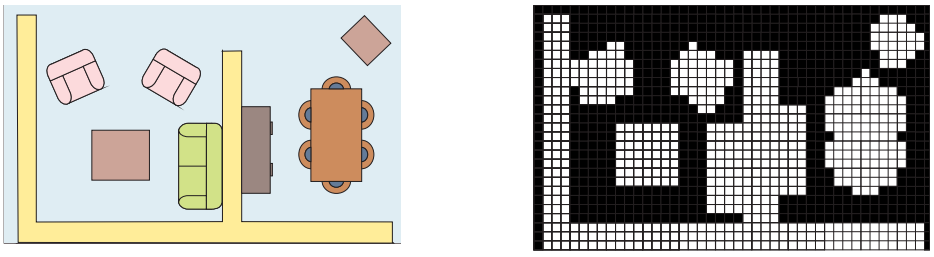
Bearing in mind the discussion in the previous chapter, let us now assume that at every time step we have access to a pose estimate (which may carry some uncertainty) as well as a range measurement of the environment relative to the robot. With this information we wish to devise a mapping algorithm that, ultimately, encapsulates uncertainty in both the measurements and the pose estimates. The occupancy grid algorithm is well suited to handle measurement uncertainty and has been widely used to that effect [18; 34; 41; 47; 109]. In Chapter 5 we introduce pose uncertainty into this algorithm.

This chapter deals with the basic occupancy grid algorithm. We provide a brief discussion on the early development of the algorithm as well as a derivation of its update equations. We also discuss some implications of an independence assumption made to lower the dimensionality of the mapping problem. This is followed by a few simple simulations to illustrate the basic idea behind the algorithm. We conclude the chapter by providing an overview of further research and extensions related to the basic occupancy grid algorithm.

3.1 Algorithm development

The first step in the occupancy grid mapping algorithm is to discretize the area to be mapped into a regular grid of cells. In 2D these cells are squares and in 3D they are cubes. For the sake of convenience, we choose to align our coordinate axes with the cells so that a single cell is fully described by its centre and a side length which, in the basic algorithm, is the same for all cells.

The second step is to consider the range measurements in an attempt to assign a probability to each cell indicating whether the cell contains an obstacle, i.e. is occupied, or has free space through which the robot may move safely, i.e. is unoccupied or free. A simple example is presented in Figure 3.1.



(a) environment (b) occupancy grid map

Figure 3.1 – A simple example to illustrate the idea behind occupancy grid mapping. An area to be mapped is depicted in (a). The light blue colour indicates free space. The corresponding 2D occupancy grid is shown in (b), where the black cells are labelled free and the white ones occupied.

All cells receive an initial probability which may correspond to "unknown" if no information about the environment is available, or may be some other value if prior information about the environment is available.

Since measurements of the environment may be corrupted by noise, and the exact pose of the robot on which the sensor is mounted may be uncertain, a hard assignment as either occupied or free might be risky. Instead it seems more appropriate to assign a probability that the cell is occupied, and herein lies the power of the occupancy grid algorithm. Once a measurement is obtained, all cells in the field of view are updated according to all available information, past and present.

The first occupancy grid algorithm was introduced by Moravec and Elfes [69]. They divide the area to be mapped into cells and build two separate grids, one indicating the probability that a cell is occupied and the other that it is free. It is assumed that range measurements are corrupted by Gaussian noise and, as measurements are received, cells are updated accordingly. A final map is built by choosing the state associated with the highest confidence for each cell. If we denote the value of cell i by map_i, the event that cell i is occupied by m_i and its complement by m_i^c, the combination of the two separate grids is given by

$$\text{map}_i = \begin{cases} p(m_i), & \text{if } p(m_i) \geq p(m_i^c), \\ -p(m_i^c), & \text{if } p(m_i) < p(m_i^c). \end{cases} \tag{3.1.1}$$

In [63] a probabilistic update equation is derived by applying Bayes' rule. To do this, conditional independence is assumed between grid cells (an assumption we also make later on). The update equation becomes

$$\text{map}_i = p(m_i|z_t) = \frac{p\left(z_t|m_i\right) p\left(m_i|z_{t-1}\right)}{p\left(z_t|m_i\right) p\left(m_i|z_{t-1}\right) + p\left(z_t|m_i^c\right) p\left(m_i^c|z_{t-1}\right)}, \tag{3.1.2}$$

where z_t represents the measurement at time t. An additional simplifying assumption is made, namely that $p\left(z_t|m_i^c\right) = 1 - p\left(z_t|m_i\right)$, which is not true in general but changes (3.1.2) to

$$p(m_i|z_t) = \frac{p\left(z_t|m_i\right) p\left(m_i|z_{t-1}\right)}{p\left(z_t|m_i\right) p\left(m_i|z_{t-1}\right) + \left(1 - p\left(z_t|m_i\right)\right)\left(1 - p\left(m_i|z_{t-1}\right)\right)}. \tag{3.1.3}$$

This update equation has the important advantage that it is associative and commutative, enabling us to incorporate measurements in any order.

The idea of expressing the update equation using Bayes' rule has been developed further by expressing the probability in its log odds form [68]. The resulting update equation forms the basis of many variants of the basic occupancy grid algorithm [17; 23; 94; 107], and is derived in the next section. We follow a procedure similar to the one in [96].

3.2 Derivation of the basic occupancy grid algorithm

As input to the system we have, from the first time step up to time t, pose estimates and measurements of the robot's environment denoted by $x_{1:t}$ and $z_{1:t}$ respectively. The measurements are obtained from a range sensor mounted on the robot, which implies that they are given relative to the robot's pose.

From the previous chapter we know that the pose estimate at a particular time step is given in the form of a probability distribution (either in closed form, or a collection of weighted particles that approximates the distribution). Since the basic occupancy grid algorithm is not equipped to handle pose uncertainty, we assume in this chapter that the pose estimate x_t at time t is the most likely pose from that distribution. In Chapter 5 we extend the algorithm in order to accommodate uncertainties in the pose estimate.

When we implement the occupancy grid algorithm we divide the environment to be mapped into a regularly spaced grid of cells. If m_i is the event that cell i is occupied, our aim is to determine the joint posterior probability distribution

$$p\left(m_1, m_2, \ldots, m_C \,|z_{1:t}, x_{1:t}\right), \tag{3.2.1}$$

where C is the number of cells.

A major problem with estimating (3.2.1) directly is its dimensionality. If we want to classify every cell as either occupied or free then, with a grid of say 1000 cells, there are $2^{1000} \approx 1.3 \times 10^{30}$ possibilities to be evaluated, which is intractable. We circumvent this problem by estimating the state of each individual cell,

$$p\left(m_i \,|z_{1:t}, x_{1:t}\right), \tag{3.2.2}$$

and assuming conditional independence between the grid cells so that

$$p(m_1, m_2, \ldots, m_C|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}). \tag{3.2.3}$$

This conditional independence assumption is discussed further in section 3.3.

If we assume that the environment is static, the discrete binary Bayes filter with static state is well suited to solve this estimation problem. We proceed to derive a recursive formula to update (3.2.3) by using the log-likelihood ratio, as this avoids truncation errors for probabilities close to 0 or 1. It also has the advantage that the multiplication of probabilities becomes the addition of log-likelihoods.

As mentioned above, measurements from the sensor are given relative to the robot's current pose. By transforming the measurement into a global coordinate system, the pose information can be incorporated into the measurement and the pose itself does not supply any additional information about the environment. Following [96] this means that we may omit the pose information $x_{1:t}$, so that

$$p(m_i|z_{1:t}, x_{1:t}) = p(m_i|z_{1:t}). \tag{3.2.4}$$

By employing Bayes' rule and assuming independence between measurements we write (3.2.4) as

$$\begin{aligned} p(m_i|z_{1:t}) &= \frac{p(z_t|m_i, z_{1:t-1})p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})} \\ &= \frac{p(z_t|m_i)p(m_i|z_{1:t-1})}{p(z_t|z_{1:t-1})}. \end{aligned} \tag{3.2.5}$$

From Bayes's rule it follows that

$$p(z_t|m_i) = \frac{p(m_i|z_t)p(z_t)}{p(m_i)}, \tag{3.2.6}$$

which after substitution into (3.2.5) produces

$$p(m_i|z_{1:t}) = \frac{p(m_i|z_t)p(z_t)p(m_i|z_{1:t-1})}{p(m_i)p(z_t|z_{1:t-1})}. \tag{3.2.7}$$

If m_i^c is the complement of m_i, i.e. the event that cell i is free, it follows similarly that

$$p(m_i^c|z_{1:t}) = \frac{p(m_i^c|z_t)p(z_t)p(m_i^c|z_{1:t-1})}{p(m_i^c)p(z_t|z_{1:t-1})}. \tag{3.2.8}$$

Division of (3.2.7) by (3.2.8) eliminates some factors that may be difficult to compute, and we obtain

$$\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})} = \frac{p(m_i|z_t)}{p(m_i^c|z_t)} \frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})} \frac{p(m_i^c)}{p(m_i)}. \tag{3.2.9}$$

Taking the natural logarithm on both sides yields

$$\log\left(\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})}\right) = \log\left(\frac{p(m_i|z_t)}{p(m_i^c|z_t)}\right) + \log\left(\frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})}\right) - \log\left(\frac{p(m_i)}{p(m_i^c)}\right). \tag{3.2.10}$$

The log odds ratio on the left hand side of equation (3.2.10) can be converted to a probability, since if

$$\lambda = \log\left(\frac{p(m_i|z_{1:t})}{1 - p(m_i|z_{1:t})}\right), \tag{3.2.11}$$

we have

$$p(m_i|z_{1:t}) = \frac{e^{\lambda}}{1+e^{\lambda}}. \tag{3.2.12}$$

If an occupancy probability is mentioned in the remainder of this thesis, it will be assumed that it can easily be converted to a log odds ratio and vice versa.

The term

$$\log\left(\frac{p(m_i)}{p(m_i^c)}\right) \tag{3.2.13}$$

is the log odds of the prior probability of the cell which, in the absence of additional information, can be set to 0 so that $p(m_i) = 0.5$. Table 3.1 lists some of these correspondences and their meanings within the occupancy grid framework.

$p(m_i)$	$\log\left(\frac{p(m_i)}{1-p(m_i)}\right)$	**Interpretation**
0	$-\infty$	definitely free
0.5	0	unknown
1	∞	definitely occupied

Table 3.1 – Correspondences between probabilities and log odds ratios as well as their respective interpretations. Here m_i is the event that cell i is occupied.

The second term on the right hand side of (3.2.10) amounts to an estimate of the occupancy probability of cell i which takes into account all previous measurements. The first term on the right hand side of (3.2.10) contains only the information at time t, which enables us to incorporate a new measurement into the current map through a simple addition.

Also note that the first two terms on the right hand side of (3.2.10) require probabilities of the form

$$p(m_i|z). \tag{3.2.14}$$

This is the inverse sensor model, so named since it is more natural to think that the state of the environment influences the measurement and not the other way around. A derivation and detailed discussion of the inverse sensor model is presented in Chapter 4.

By introducing the notation $l_{i,t}$ which denotes the log odds ratio of cell i at time t, the update equation (3.2.10) becomes

$$l_{i,1:t} = l_{i,t} + l_{i,1:t-1} - l_0. \tag{3.2.15}$$

Here the prior is denoted by l_0 which, in the absence of prior information, is the same for all cells.

An additional advantage of the update equation in this form is that, if an occupied cell is observed numerous times, its occupancy probability increases automatically. In fact, if a noise-free sensor were to observe an occupied cell an infinite number of times, its occupancy probability would become exactly one, which is a desirable quality of an effective mapping algorithm. Similarly, the probability would tend to zero if the cell was unoccupied.

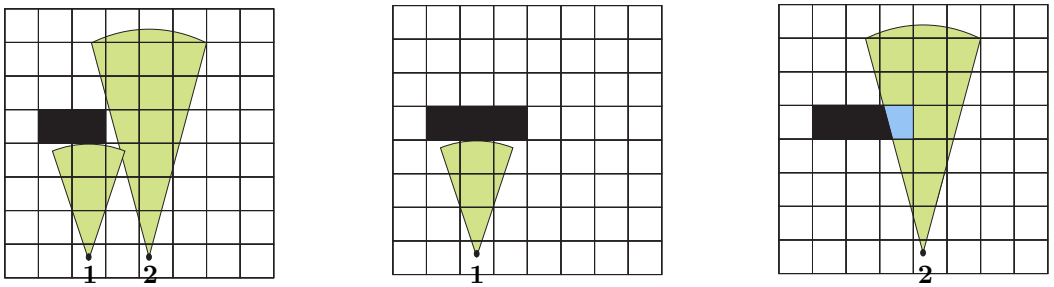
3.3 The independence assumption

In order to derive the basic occupancy grid algorithm we assume conditional independence between grid cells, i.e.

$$p(m_1, m_2, \ldots, m_C|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}). \tag{3.3.1}$$

This is a strong assumption and one should examine the consequences of this assumption.

Consider the scenario depicted in Figure 3.2. Here a single beam of a noise-free sensor takes two measurements of the environment at distinct times, as shown in (a). The two beams are indicated in green, while the black and white cells indicate occupied and free cells respectively. Beam 1 returns a measurement corresponding to an obstacle somewhere in that beam and, since the beam stretches across three cells, all three have their occupancy probabilities increased. Beam 2, however, returns maximum range and indicates that one of those three cells is free, resulting in a conflict which is indicated by the blue region in (c). This conflict arises as a direct consequence of the independence assumption. Fortunately, as the algorithm adds log odds ratios, the conflict results in a value close to zero which is treated cautiously during navigation as it corresponds to unknown. From this example we see that, if the width of a sensor beam stretches over multiple cells, dependencies between neighbouring cells are introduced. If we assume conditional independence between grid cells, we are unable to model these dependencies [96].



(a) environment (b) measurement 1 (c) measurement 2

Figure 3.2 – Investigation of an effect of the independence assumption. White cells represent free areas and black occupied regions. In (a) a single beam of a noise-free sensor gathers two consecutive measurements of the environment. Here the cell colours reflect the true states. The measurement from beam 1 results in the occupancy grid shown in (b), since that beam stretches over three cells. The measurement from beam 2 indicates that one of those cells is free, resulting in a conflict indicated by the blue region in (c).

If the width of a sensor beam never exceeds the width of a single cell, this situation is avoided. In this work we consider only sensors with narrow beams (typically less than 1°). For short range sensors with a maximum effective range of say 5 m, the arc length at maximum range corresponds to about 8.7 cm. This implies that if the side length of the grid cells are more than 8.7 cm, the problem is avoided. This resolution depends on the maximum effective range and the width of the beams. A more detailed discussion on the maximum effective range is given in Chapter 4. For a typical laser range scanner, the beam width is about 0.5° resulting in a resolution of about 4.36 cm for a maximum range of 5 m. For the same maximum range one pixel of a CCD camera sensor may translate to less than 10 mm (depending on the intrinsic parameters). These resolutions seem feasible for general navigation purposes [76]. If a sensor with a wider beam is required, or the minimum resolution is inadequate for the application, the alternative is to be aware of such conflicts that may arise or to implement an algorithm that maintains dependencies between grid cells. One such approach is mentioned in section 3.5.4.

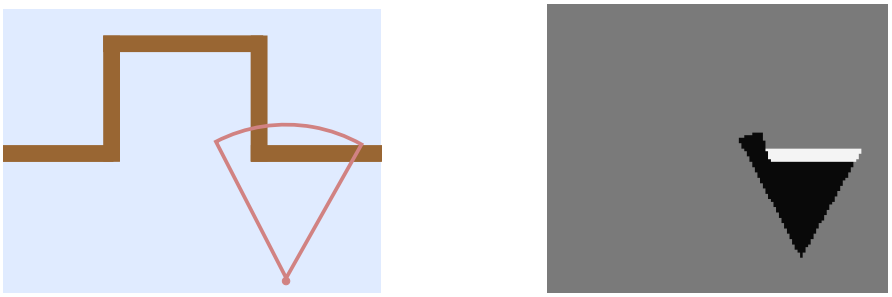
3.4 Examples

In this section we provide some simulated examples to illustrate the basic occupancy grid algorithm. Similar examples are presented in Chapters 4 to 6 to illustrate every new component we explain.

In the update equations the inverse sensor model is required, which we have not yet explained. For the moment we assume a noise-free sensor. Ideally, we prefer to assign a probability of 1 to cells that, according to the sensor, contain obstacles, a value of 0 to cells that are seen by the sensor but are unoccupied and a value of 0.5 to all cells that are not seen by the sensor. When we convert probabilities of 1 and 0 to their log odds ratios, they become ∞ and $-\infty$ respectively, which means that no subsequent updates will be able to change these values. For this reason we truncate probabilities close to 0 and 1, but more details are given in Chapter 4.

This noise-free inverse sensor model is unrealistic since measurements are often corrupted by noise. In Chapter 4 we discuss how we incorporate measurement uncertainty into the inverse sensor model.

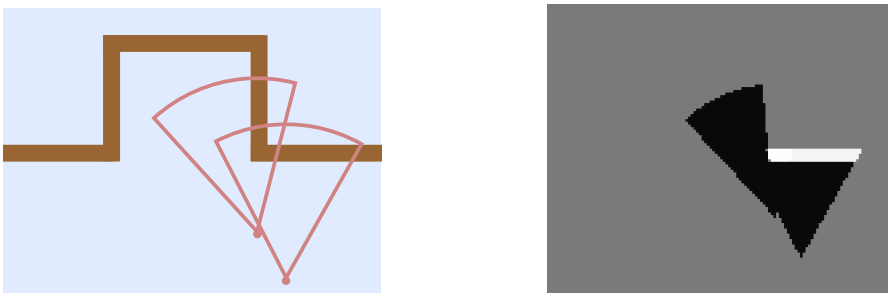
In Figure 3.3(a) an environment and a single field of view is shown. Obstacles are shown in brown and the pink dot indicates the location of the sensor. The resulting grid map can be seen in Figure 3.3(b), where occupancy probabilities are indicated in shades of grey (the lighter the colour of a cell, the higher the occupancy probability). In this example no noise is present in the measurement or the pose, so the occupancy grid should agree with the environment.



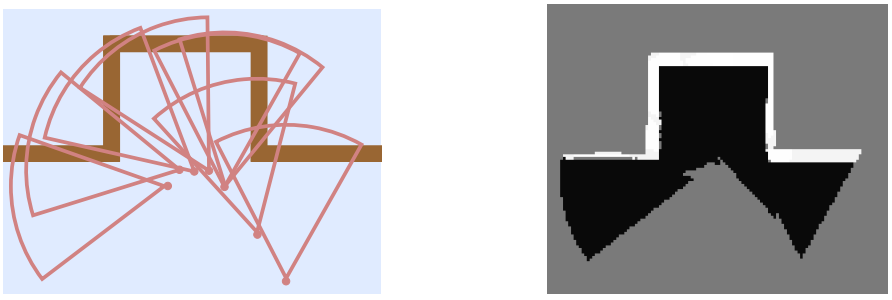
(a) environment and field of view (b) occupancy grid map

Figure 3.3 – A simulated example of a 2D occupancy grid map. In (a) the environment is shown, where brown regions represent obstacles. A field of view is also indicated. The occupancy grid resulting from this measurement is shown in (b). The larger a cell's occupancy probability, the lighter the colour.

At a subsequent time step, one would expect the occupancy probabilities of overlapping regions to be reinforced by the new measurement. Two fields of view are shown in Figure 3.4(a) and the occupancy grid after these two measurements are shown in (b).



(a) two measurements (b) occupancy grid map



(c) eight measurements (d) occupancy grid map

Figure 3.4 – The occupancy grid maps after two and eight measurements are incorporated.

The corner in Figure 3.4(a) is observed in both measurements and therefore has a higher occupancy probability than other parts of the obstacle, which are observed only once. Figure 3.4(c) shows eight fields of view and (d) shows the occupancy grid map after incorporating these measurements.

3.5 Overview of related work

Since it was first introduced in 1985, much research has been conducted on the occupancy grid algorithm and various improvements have been made. In this section we provide a brief overview of some of these approaches.

3.5.1 General improvements

When a sensor receives a measurement, we assume that it provides the location of an object along a particular line of sight. The observed obstacle may not be the only one that occurs in this line of sight, but is merely the first observed one. In [57] the idea of multiple targets as opposed to single target models is introduced. The idea is that cells further away than the range measurement should not have their occupancy probabilities lowered. We also incorporate this idea into our algorithm's capacity to handle measurement uncertainty, as will be explained in Chapter 4.

Although the occupancy grid was initially introduced in 2D, it has since been extended to 3D applications [48; 75; 104]. Adding another dimension has no influence on the update equations but may increase computation since considerably more cells have to be updated.

3.5.2 Dynamic environments

In some applications occupancy grids are required for dynamic environments [13; 65; 76]. In [108] the authors discuss the map overconfidence problem of the occupancy grid update equation in its basic form. If a sensor observes part of the map for a long period of time and the state of this part changes, e.g. a parked car is removed from a parking space, it requires roughly the same number of measurements to bring the occupancy probability down to unknown and even more to alter the state to free space. This means that the update equation is unsuitable for dynamic environments. Instead they propose an exponential "forgetting" policy to solve this problem.

Another approach for modelling dynamic environments within the occupancy grid paradigm is to represent obstacles as cells with size and speed. The authors of [32] propose tracking obstacles within an occupancy grid using a particle filter. The particles are able to move between cells from one time step to the next and provide information on the position and speed of the obstacles.

In this work, however, we assume a static environment and regard these extensions to be beyond the scope of the thesis.

3.5.3 Maximum a posteriori mapping

If a situation arises where all the poses and measurements over time are available simultaneously, i.e. in a post-processing paradigm, the general procedure for building a map of the environment is the maximum a posteriori (MAP) approach. The estimation problem becomes

$$m^* = \arg\max_m \ \log p(m_1, m_2, \ldots, m_C | z_{1:t}, x_{1:t}). \tag{3.5.1}$$

It can be shown [96] that this leads to

$$m^* = \arg\max_m \ \left[\sum_t \log p(z_t | x_t, m_1, m_2, \ldots, m_C) + \sum_i m_i \log \left(\frac{p(m_i)}{p(m_i^c)} \right) \right], \tag{3.5.2}$$

which can be solved by hill-climbing. An example of such an algorithm would start with all cells labelled free and flip a cell to occupied when such a flip increases the likelihood of the data. No independence assumption between grid cells has been made, which is a significant advantage of this method. The drawback of this approach is that all information has to be available for the estimation and a hard assignment is made, whereas a probabilistic map is more desirable for path planning and obstacle avoidance. This approach is not followed in our work, since we desire an incremental solution.

3.5.4 Forward sensor models

The inverse sensor model, introduced in the derivation of the algorithm in section 3.2, is given by

$$p(m_i|z_t), \tag{3.5.3}$$

where m_i is the event that cell i is occupied. Conversely, the probability

$$p(z_t|m_i) \tag{3.5.4}$$

is called the forward sensor model. In [92] the forward sensor model is used to solve the occupancy grid mapping problem in the original high-dimensional space and to maintain dependencies between neighbouring cells.

The forward sensor model is specified as a Gaussian mixture model with latent variables corresponding to three different measurement causes: random noise, from an actual obstacle and corrupted with assumed Gaussian noise, or the maximum range in which case obstacles along the ray of sight may be completely missed. An expected log-likelihood expression is obtained and the EM algorithm is employed to search for maps that maximize the likelihood of the measurements.

A disadvantage of this approach is that it keeps no notion of uncertainty in the map. Moreover, a hard assignment is made, i.e. a cell is either occupied or free, and the map is therefore not probabilistic. Another significant disadvantage is the fact that it cannot be implemented incrementally and has to perform many iterations for convergence.

The forward model approach proposed in [75] introduces similar intermediate variables, but does not require the EM algorithm for cell updates. However, these forward models are determined experimentally and may be hard to come by [70].

For these reasons we do not employ the forward model in our approach and, instead, proceed in the next chapter with an investigation of the inverse sensor model.

Chapter 4

Measurement uncertainty

From the previous chapter we know that each cell in an occupancy grid must be assigned a probability value based on the measurements captured at discrete time steps. A measurement consists of one or multiple rays in space, each associated with an indication of distance to the first observed obstacle along that ray. All cells intersected by a ray must be updated according to some function of that distance, and the rules specifying these updates are known as measurement models.

Since measurements in practice are corrupted by noise, it can be beneficial to include measurement uncertainty in the occupancy grid mapping algorithm. Sensor models are especially amenable to incorporate measurement uncertainty and can be even further refined to include sensor-specific characteristics.

The update equation derived in the previous chapter requires the probability $p(m_i|z_t)$. This is known as the inverse sensor model, because it is the true state of cell i that influences the measurement z_t, contrary to what the expression suggests. In this chapter we derive an inverse sensor model analytically and investigate how to refine the model for various types of sensors.

4.1 Assumptions

In order to proceed to the derivation of our inverse sensor model, a few important assumptions are made.

We assume that the range sensor provides a set of coordinates, each of the form (θ, r) where θ is the ray angle and r the distance from the sensor to the first observed obstacle. This is a reasonable assumption because the output of some sensors, such as laser range scanners, are indeed in this form. If the output is an image with pixel values, as is typically the case with stereo cameras, it can be transformed into the desired form. Three-dimensional sensors generally provide obstacle coordinates of the form (θ, ϕ, r) and are therefore also suitable for our applications.

Furthermore, we concentrate on sensors that have fairly narrow beams, each of which spanning less than 1° and returning a single measured distance. The uncertainty in angle is then virtually negligible in comparison to the uncertainty in distance (along r). By doing so we are able to employ one-dimensional inverse sensor models which are functions exclusively of the distance r.

We also assume independence between measurement rays so that we may handle the updating process separately for each ray.

4.2 Analytical derivation of an inverse sensor model

In this section we derive an analytical inverse sensor model by assuming the measurement is corrupted by Gaussian noise. In order to do so, we first consider the ideal case in which the sensor is noise-free.

4.2.1 Ideal inverse sensor model

A range measurement consists of multiple rays in space. Each of these rays is handled separately and all cells intersected by a ray should receive an updated probability value depending on the recorded distance. We first discuss the form of this update profile for all cells along a particular ray assuming, for the moment, that the sensor is noise-free.

When a new measurement is received at time t, a cell may already have a log odds ratio from previous measurements. The inverse sensor model is used to calculate a new log odds ratio based on the new measurement, which is added to the current log odds ratio. Here we are concerned with calculating this new log odds ratio.

Suppose for a specific ray with angle θ (or angle pair in the case of 3D measurements), a measurement $r = Z$ is received at time t. This means that, according to the sensor, the first obstacle along that ray of sight occurs at a distance Z. The ideal inverse sensor model should return a probability value corresponding to certainly free ($p(m_i|z_t) = 0$) to all cells in front of the measurement distance Z, a value corresponding to certainly occupied ($p(m_i|z_t) = 1$) to the cell containing the obstacle, and a value corresponding to unknown ($p(m_i|z_t) = 0.5$) to all cells further away than Z since the first obstacle may not be the only one along that ray. An example of this ideal sensor model is shown in Figure 4.1, where a measurement of $Z = 2$ was recorded and the resulting probability as a function of the cell distance r from the sensor is plotted.

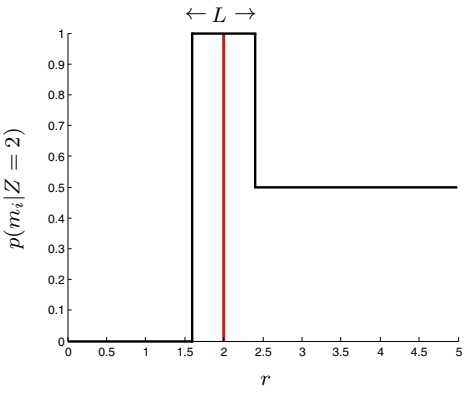


Figure 4.1 – The ideal inverse sensor model for a measurement at $Z = 2$. Note that we do not assign a value of 1 exclusively to $r = 2$, but to a band of width L surrounding $r = 2$. This is to ensure that the cell in which the obstacle falls receives a value of 1.

The centre of a cell is used to calculate the distance to the sensor and this distance is used to evaluate the sensor model for an update value. However, the peak at Z may be completely missed if the centre of the cell has a slight offset from Z. Alternatively, we can find the average value over all possible distances that fall inside the boundary of the cell, using integration, but because we deal with many cells and many measurement beams this approach is computationally too expensive. Instead we introduce a parameter L, signifying a band of r values that receive a probability corresponding to certainly occupied, as shown in Figure 4.1. A natural choice for the value of this parameter is the diagonal distance between two corners of a grid cell.

We prefer to adapt the ideal sensor model to incorporate measurement uncertainty. Such inverse sensor models can be obtained through training or theoretical derivation, although many authors merely depict their models graphically and provide no explicit equations [63; 68; 108]. In the next section we derive an analytical expression for the inverse sensor model with Gaussian noise.

4.2.2 Inverse sensor model with Gaussian noise

In practice sensor measurements may be noisy. For the sake of convenience, we assume the noise to be normally distributed around the observed value Z, with a standard deviation σ that may depend on Z, so that

$$r \sim \mathcal{N}\left(Z, \sigma^2\right). \tag{4.2.1}$$

The probability distribution function (pdf) of this Gaussian is given by

$$f(r; Z, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r-Z)^2}{2\sigma^2}}, \tag{4.2.2}$$

and is depicted in Figure 4.2. Note that σ is an indication of how certain we are that the observed obstacle is actually located at Z. A larger σ widens the peak and increases our uncertainty, while a smaller σ results in a narrower peak and decreases our uncertainty.

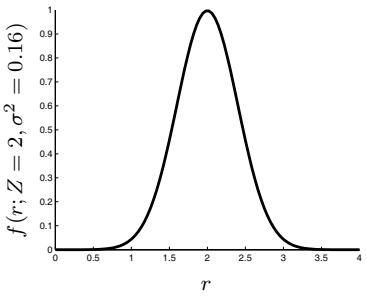


Figure 4.2 – The probability distribution of a measurement at $Z = 2$ corrupted by Gaussian noise with standard deviation $\sigma = 0.4$.

In order to translate this noise model to the ideal sensor model, a convolution between the functions in Figures 4.1 and 4.2 is performed. The function in Figure 4.1, however, is defined piecewise so special care has to be taken. The convolution can be performed numerically, although an analytical expression is preferred, especially when grid mapping is employed with variable grid size (as in Chapter 6).

The convolution between two real-valued functions is defined as

$$(f * g)(r) = \int_{-\infty}^{\infty} g(\tau) f(r - \tau)\, d\tau. \tag{4.2.3}$$

In order for (4.2.3) to be well defined, f or g should decay rapidly near the endpoints of its interval, which is the case for the Gaussian pdf.

We define the function $g(r)$, which represents the ideal sensor model, as

$$g(r) = \begin{cases} 0, & \text{if } r < Z - \frac{L}{2}, \\ 1, & \text{if } Z - \frac{L}{2} \leq r < Z + \frac{L}{2}, \\ 0.5, & \text{otherwise.} \end{cases} \tag{4.2.4}$$

It should be noted that our definition of $g(r) = 0$ for $r < 0$ is done merely for convenience in the convolution, even though it does not make sense from a practical point of view (the mapping algorithm should not assign free space to regions behind the sensor). However, when we evaluate the convolution in order to update the value of a particular cell, we will only ever do so for $r > 0$.

Since $g(r)$ is a piecewise defined function, the convolution will also be defined piecewise and therefore the integrals must be split whenever $g(r)$ changes value.

To simplify later steps, we first compute

$$F(a, b) = \frac{1}{\sqrt{2\pi}\sigma} \int_a^b e^{-\frac{(r-\tau-Z)^2}{2\sigma^2}}\, d\tau, \tag{4.2.5}$$

because the convolution can be written as

$$(f * g)(r) = kF(a, b), \tag{4.2.6}$$

where $k \in \{0, 0.5, 1\}$, depending on the interval (a, b) on which r is defined. By letting

$$u = \frac{r - \tau - Z}{\sqrt{2}\sigma}, \tag{4.2.7}$$

(4.2.5) becomes

$$F(a,b) = \frac{-1}{\sqrt{\pi}} \int_{\frac{r-a-Z}{\sqrt{2}\sigma}}^{\frac{r-b-Z}{\sqrt{2}\sigma}} e^{-u^2}\, du, \tag{4.2.8}$$

which implies that

$$F(a,b) = -\frac{1}{2}\text{erf}\left(\frac{r-b-Z}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r-a-Z}{\sqrt{2}\sigma}\right), \tag{4.2.9}$$

where $\text{erf}(x)$ is the error function defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt. \tag{4.2.10}$$

Furthermore, if a is set to $-\infty$, the definition of $F(a,b)$ changes to

$$\begin{aligned} F(-\infty,b) &= \lim_{a\to-\infty} \frac{1}{\sqrt{2\pi}\sigma} \int_a^b e^{-\frac{(r-\tau-Z)^2}{2\sigma^2}}\, d\tau \\ &= -\frac{1}{2}\text{erf}\left(\frac{r-b-Z}{\sqrt{2}\sigma}\right) - \frac{1}{2}. \end{aligned} \tag{4.2.11}$$

We proceed to compute the convolution on the respective intervals. Note that $g(r)$ is zero for $r < 0$ and the first point at which $g(r)$ changes value is at $r = Z - \frac{L}{2}$. So for $r \in \left(-\infty, Z - \frac{L}{2}\right)$,

$$\begin{aligned} (f*g)(r) &= 0\,\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{r} e^{-\frac{(r-Z)^2}{2\sigma^2}}\, d\tau \\ &= 0\,F(-\infty, r) \\ &= 0. \end{aligned} \tag{4.2.12}$$

The next change in $g(r)$ occurs at $r = Z + \frac{L}{2}$, so that for $r \in \left[Z - \frac{L}{2}, Z + \frac{L}{2}\right)$ we have

$$\begin{aligned} (f*g)(r) &= 0\,\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{Z-\frac{L}{2}} e^{-\frac{(r-Z)^2}{2\sigma^2}}\, d\tau + \frac{1}{\sqrt{2\pi}\sigma} \int_{Z-\frac{L}{2}}^{r} e^{-\frac{(r-Z)^2}{2\sigma^2}}\, d\tau \\ &= 0\,F\left(-\infty, Z-\frac{L}{2}\right) + 1\,F\left(Z-\frac{L}{2}, r\right) \\ &= -\frac{1}{2}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r-2Z+\frac{L}{2}}{\sqrt{2}\sigma}\right). \end{aligned} \tag{4.2.13}$$

For the remaining interval, $r \in \left[Z + \frac{L}{2}, \infty\right)$, the convolution is

$$\begin{aligned} (f*g)(r) &= 0\,F\left(-\infty, Z-\frac{L}{2}\right) + \frac{1}{\sqrt{2\pi}\sigma} \int_{Z-\frac{L}{2}}^{Z+\frac{L}{2}} e^{-\frac{(r-Z)^2}{2\sigma^2}}\, d\tau + \frac{0.5}{\sqrt{2\pi}\sigma} \int_{Z+\frac{L}{2}}^{r} e^{-\frac{(r-Z)^2}{2\sigma^2}}\, d\tau \\ &= 0\,F\left(-\infty, Z-\frac{L}{2}\right) + 1\,F\left(Z-\frac{L}{2}, Z+\frac{L}{2}\right) + 0.5\,F\left(Z+\frac{L}{2}, r\right) \\ &= -\frac{1}{4}\text{erf}\left(\frac{r-2Z-\frac{L}{2}}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r-2Z+\frac{L}{2}}{\sqrt{2}\sigma}\right) - \frac{1}{4}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right), \end{aligned} \tag{4.2.14}$$

which tends to 0.5 as $r \to \infty$.

An example of such a convolved function can be seen in Figure 4.3. Note that r values in front of the measurement still receive a probability corresponding to free but the function gradually increases as we move closer to the observed measurement until a peak is reached, and it tapers off behind the measurement to unknown.

A similar convolution procedure can be followed for a different noise distribution since the occupancy grid algorithm itself is not restricted to Gaussian noise.

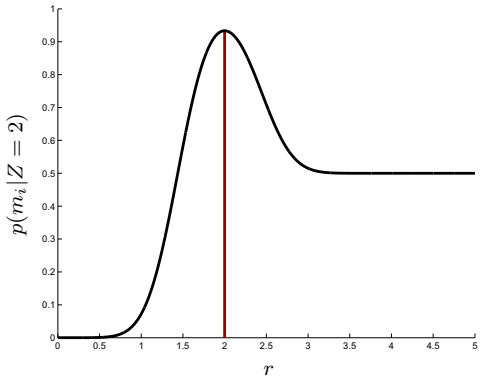
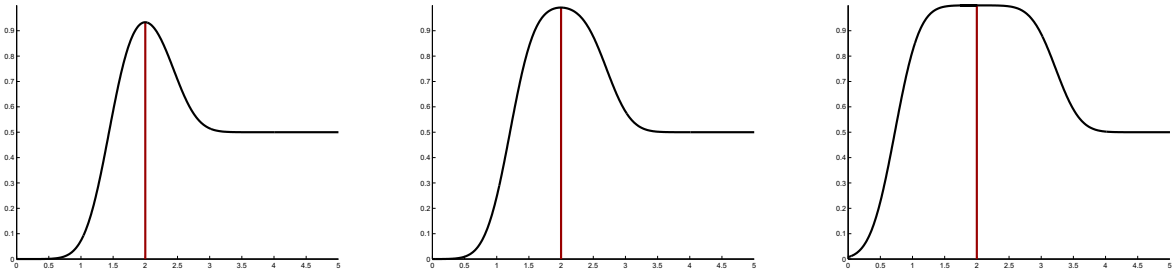


Figure 4.3 – Our Gaussian inverse sensor model for $Z = 2$. This sensor model is acquired through a convolution of the ideal sensor model and a Gaussian distribution with mean at $r = 2$ and $\sigma = 0.3$.

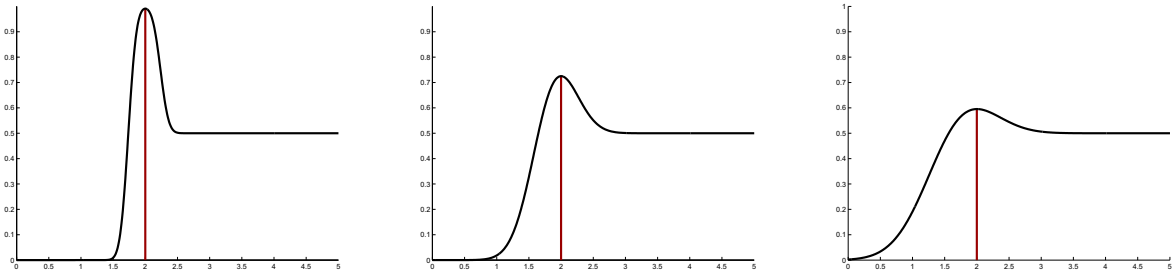
Let us consider the parameters that have an influence on the shape of the Gaussian inverse sensor model. Firstly, the value of L affects the width and the height of the peak of the inverse sensor model. The larger L, the wider and higher the peak. This can be seen in Figure 4.4.



(a) $L = 1$ (b) $L = 1.5$ (c) $L = 2.5$

Figure 4.4 – The influence of L on our Gaussian inverse sensor model. The larger the L value, the wider and higher the peak.

In Figures 4.3 and 4.4 we note that the peak may not have a value of 1 which signifies certainly occupied. A larger standard deviation σ should widen the peak and lower the maximum. Conversely, we would expect a smaller σ to result in a narrower peak at a higher maximum. Figure 4.5 shows the influence of the parameter σ on the model, which is as desired.



(a) $\sigma = 0.1$ (b) $\sigma = 0.3$ (c) $\sigma = 0.5$

Figure 4.5 – The influence of σ on our Gaussian inverse sensor model. The larger the σ value, the wider and lower the peak.

This σ value, which is linked to the measurement uncertainty, may depend on the specific sensor used and the value of Z. For example, the uncertainty in the 3D coordinates calculated from disparities for stereo cameras increases quadratically with distance [51]. Here we uncover another advantage of using the Gaussian inverse sensor model. If a function can be determined that relates the measurement uncertainty σ to the measurement Z, the Gaussian sensor approach enables us to engineer an inverse sensor model for the specific sensor employed. We explore this possibility in section 4.3.

4.2.3 Assigning a probability value to each cell

The occupancy grid algorithm updates each cell by adding the quantity

$$\log\left(\frac{p(m_i|z_t)}{1-p(m_i|z_t)}\right) \tag{4.2.15}$$

to the current value and subtracting the prior. This means that the inverse sensor model as it appears in Figure 4.3 is not used in that form, but in its log odds ratio. A value of 0 is mapped to $-\infty$ and 1 to ∞, implying that subsequent measurements are not able to alter these values, which is undesirable considering they may be incorrect. To sidestep this problem we truncate the minimum value of the inverse sensor model at a predefined value p_{free} and also do not allow any values above a predefined value of p_{occ}.

Since we handle each measurement ray separately, and it is possible for multiple rays to intersect a single cell, a cell may receive multiple distinct log odds ratios. However, a cell must be updated with a single value. We decide on the following rules for a conservative system in which we would prefer to avoid collisions with obstacles.

- If all log odds ratios are negative, choose the minimum.
- If all log odds ratios are positive, choose the maximum.
- If some of the log odds ratios are positive and some are negative, choose the maximum.

In the first of these cases all measurements agree that the cell is free. Since we do not want to increase uncertainty unnecessarily, we choose the measurement that gives the highest confidence in the cell being free. Similarly, if all measurements agree that the cell is occupied, we choose the maximum value. If, however, the cell receives mixed measurements we choose to be conservative by assigning a value corresponding to occupied, since we prefer to avoid obstacles that may not exist rather than to collide with an unmapped one.

Now that we have introduced the inverse sensor model and explained how it is utilized in the updating process, we proceed to design such models for some specific sensors.

4.3 Inverse sensor models for specific sensors

In the previous section we introduced the Gaussian inverse sensor model as a means to incorporate measurement uncertainty into the occupancy grid mapping process. In this section we look at how such a model can be engineered to encapsulate the noise associated with a specific sensor. We consider types of sensors typically used for robotic mapping. A good understanding of the uncertainty in the measurement is important not only in the eventual mapping process, but also to build realistic simulation environments in which to test our algorithm.

The Gaussian distribution used in the previous section is described by a mean Z, and a standard deviation σ which may be a function of Z if, for example, the measurement uncertainty of a particular sensor increases with distance. The effect of this on the Gaussian sensor model is that the peak widens and its height decreases, as can be seen in Figure 4.6 where, for clarity, the Z value used to generate each curve is shown in red. This behaviour is as expected [2; 57].

By making certain assumptions about the accuracy with which a sensor returns a measurement we can, in many cases, derive a function $\sigma(Z)$ using the so-called delta method.

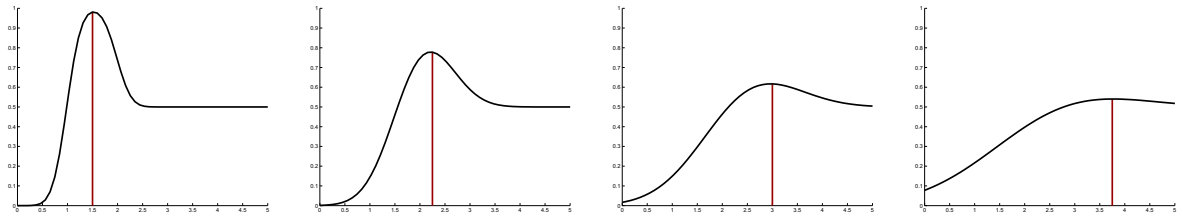


Figure 4.6 – The effect of a distance dependent standard deviation on our Gaussian inverse sensor model. The Z value used to generate each curve is shown in red.

4.3.1 The delta method

Suppose that a sensor provides, for a single measurement beam, an output variable d that can be transformed to a range distance according to

$$r = r(d). \tag{4.3.1}$$

If we assume that the true value d^*, which corresponds to the true range distance $r(d^*)$, is normally distributed around the measured value d and has standard deviation σ_d, we may write

$$d^* \sim \mathcal{N}\left(d, \sigma_d^2\right). \tag{4.3.2}$$

Since the output d is transformed using the function $r(d)$, we are interested in the distribution of r. According to the delta method, the transformed distribution can be approximated by a Gaussian [102], such that

$$r \sim \mathcal{N}\left(r(d), \sigma_d^2 \left[r'(d)\right]^2\right), \tag{4.3.3}$$

which is equivalent to

$$r \sim \mathcal{N}\left(Z, \left[\sigma(Z)\right]^2\right), \tag{4.3.4}$$

where r' denotes the first derivative of r and we have defined $Z = r(d)$ and $\sigma(Z) = \sigma_d r'(d)$.

Since the delta method employs a truncated Taylor expansion, this approximation is only useful if the measured output d has a high probability of being close to the true value d^*, or if the function $r(d)$ is linear in d (which is rarely the case).

Keeping this in mind, we proceed to find the functions $Z = r(d)$ and $\sigma(Z)$ for some specific sensors in order to build inverse sensor models for them.

4.3.2 Stereo camera sensors

A stereo camera rig consists of two synchronized cameras mounted at a fixed distance from each other. The output at every time step is a set of two images. Pixels are matched between left and right images and, if the images are rectified [50], those matches can be transformed to 3D coordinates. If (x_L, y_L) is the pixel coordinates of a feature in the left image and d is the disparity between this pixel and its match in the right image, the transformation to 3D is given by

$$\begin{bmatrix} X_w W \\ Y_w W \\ Z_w W \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{B} & 0 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ d \\ 1 \end{bmatrix}, \tag{4.3.5}$$

where B is the baseline distance between the two cameras, X_w, Y_w, Z_w represent the 3D location of the feature relative to the sensor, and (p_x, p_y) and f are intrinsic camera parameters (obtained from a calibration of the system) representing the principal point coordinates and focal length respectively.

The distance from the feature to the sensor is determined as

$$r = \sqrt{X_w^2 + Y_w^2 + Z_w^2}. \tag{4.3.6}$$

Substituting the right hand side of (4.3.5) into (4.3.6) produces

$$r(d) = \frac{B\omega}{d}, \tag{4.3.7}$$

where

$$\omega = \sqrt{(x_L - p_x)^2 + (y_L - p_y)^2 + f^2}. \tag{4.3.8}$$

If we assume that the true disparity d^* is

$$d^* \sim \mathcal{N}(d, 1), \tag{4.3.9}$$

i.e. normally distributed about the measured disparity d with a standard deviation of one pixel, it follows from (4.3.7) that

$$r'(d) = -\frac{B\omega}{d^2} = -\frac{r(d)^2}{B\omega}, \tag{4.3.10}$$

which exists and is nonzero for all $d > 0$ if $B > 0$ and $f > 0$. In practice both B and f are indeed positive, and $d = 0$ only for features that are infinitely far from the sensor. According to the theory in section 4.3.1,

$$r \sim \mathcal{N}\left(r(d), \left(\frac{r(d)^2}{B\omega}\right)^2\right), \tag{4.3.11}$$

which implies that for a stero camera sensor the uncertainty as a function of distance Z is given by

$$\sigma(Z) = \frac{Z^2}{B\omega}. \tag{4.3.12}$$

We deduce that the uncertainty increases more or less quadratically with range.

In [2] and [51] it is claimed that a small error Δd in disparity results in an error in distance ΔZ, where

$$\Delta Z = \frac{Z^2}{Bf}\Delta d, \tag{4.3.13}$$

which is the same as our formula if the small error in disparity is set to 1 and $\omega = f$, implying that $x_L = p_x$ and $y_L = p_y$. It can be sufficient to compute the uncertainty for this special case since, for these parameters, ω is a minimum and therefore implies that for a fixed range distance Z, the uncertainty is a maximum.

Figure 4.7 depicts a top-down view of feature locations along with their associated uncertainties. The stereo cameras are indicated by the black boxes on the x-axis. For the lines in blue, the measurements returned by all the beams are $Z = 2$. Uncertainties are calculated according to (4.3.12) and are reflected by the lengths of the blue line segments. The green segments and dots correspond to measurements $Z = 1.4$. Note that the uncertainty is at a maximum in front of the cameras. Since each pixel captures information in the beam that connects the camera centre to the pixel boundaries, and since an image consists of regularly spaced pixels, the beams are narrower near the side of the image which explains this maximum uncertainty at the image centre.

By considering (4.3.12) one can decide on a maximum effective range $Z_{\max}$, and ignore all measurements beyond this distance (by setting their values to $Z_{\max}$) due to their large uncertainty.

4.3.3 Structured light sensors

A structured light sensor is equipped with a projector and a camera, located at a fixed distance from each other. The projector projects a pattern onto the scene and the camera captures the resulting distortion. From this distortion depth information of the scene is inferred.

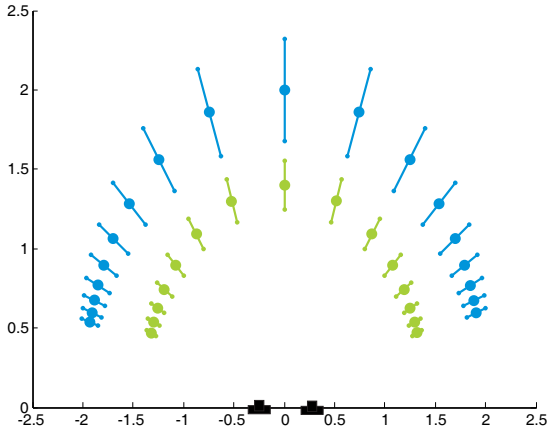


Figure 4.7 – A top-down view of feature locations and their associated uncertainties along specific rays for a pair of stereo cameras (indicated in black on the x-axis). Two distinct range measurements are assigned to all the rays, namely $Z = 2$ (blue) and $Z = 1.4$ (green). The lengths of the line segments indicate the uncertainties associated with the measurements.

This type of sensor is closely related to stereo cameras because the projector can be thought of as a second camera, and range is also determined through triangulation. For this reason the uncertainty will be of the same form as that of stereo cameras, i.e.

$$\sigma(Z) = \frac{Z^2}{Bf}, \tag{4.3.14}$$

where Z is the range measurement, B is the baseline between the projector and the camera and f is the focal length. The uncertainty given in [9] for a triangulation-based laser scanner is of a similar form.

Since the uncertainty of triangulation-based sensors increases with range squared, they are generally employed as short range scanners (less than $10m$).

4.3.4 Time-of-flight sensors

Sensors returning long range measurements (exceeding $10m$) usually employ time-of-flight technology [9]. These sensors, called LIDAR (light detection and ranging), emit short impulses of near infrared light in a narrow beam along a specific direction and measure the time it takes for the signal to reach a reflective surface and travel back to the sensor. Distance is calculated as the product of speed and time, implying that

$$Z = \frac{v\tau}{2}, \tag{4.3.15}$$

where v is the wave propagation velocity and τ is the measured time for the round trip. Such a sensor may be equipped with a rotating mirror that enables it to start at a certain point and rotate through multiple angles, all the while emitting light impulses and measuring the time delay for each angle so that a semi-circular area around the sensor is mapped. This means that such a measurement is not made instantaneous, which should be accounted for when synchronizing the mapping sensors with the SLAM sensors.

One advantage of employing time-of-flight sensors is that it is possible to measure the round trips through different media such as air, water, glass, etc., by adjusting v according to

$$v = \frac{c}{n}, \tag{4.3.16}$$

where c is the speed of light in a vacuum and n is the refractive index, a constant depending on the medium. For air $n \approx 1.0003$.

One factor that may influence the uncertainty is the pulse rise time which is an indication of the time it takes for a signal, such as the emitted light wave, to change from a low value to a high value. Another

factor is the signal to noise ratio (SNR), which is the ratio between the useful information (the signal) and the unwanted noise. A high SNR is therefore desirable but it depends on the distance measurement, environmental factors, as well as on the detection mechanism. According to [9] the associated uncertainty with this type of sensor is given by

$$\sigma = \frac{vT_r}{2\sqrt{\text{SNR}}}, \tag{4.3.17}$$

where T_r is the pulse rise time, which is constant for a specific wavelength. Although the SNR depends on the noise of the environment, according to [9] a typical value is SNR = 100 (for more information on the SNR for laser range scanners, the reader is referred to [15]). The uncertainty may then amount to 0.0001% of the range measurement, so many assume a constant measurement uncertainty [14; 66; 87; 88] based on the above mentioned characteristics.

A complication for time-of-flight sensors is that specular readings may occur. This happens when the surface makes an oblique angle with the beam axis and the signal is not reflected back to the sensor directly, but bounces off multiple other surfaces before it reaches the transmitter. For more information on how to incorporate this behaviour into a sensor model, the reader is referred to [57].

4.3.5 Other sensors

Sonar sensors also use time-of-flight technology by emitting sound rather than light. Because of atmospheric disturbances, these sound waves may be severely corrupted with noise and may be inaccurate over long ranges (exceeding $10m$).

The long range radar (LRR) sensor detects moving objects with high precision by emitting an electromagnetic wave and calculating its time-of-flight. It usually returns a single measurement for a wide beam, and the output is in the form of the polar coordinates of a detected object and its Doppler speed.

Such a sensor typically has one wide beam that returns a single measurement [69], so that the measurement uncertainty is a function of range as well as the angle between the beam axis and the cell. Uncertainties of this kind are beyond the scope of our work, but information on how to obtain a 2D inverse sensor model may be found in [77] and [44] for sonar and LRR sensors respectively.

4.4 Training inverse sensor models

Another way of acquiring an inverse sensor model that incorporates sensor-specific characteristics is by learning [96]. Such an algorithm generates samples from the forward measurement model $p(z_t|m_i)$ and approximates the inverse using a supervised learning algorithm such as logistic regression or neural networks [89]. A significant disadvantage of such approaches is that a ground truth map is required for training. For more information on the training of inverse sensor models, the reader is referred to [96].

4.5 Sensor fusion

Since different sensors have different strengths and weaknesses, and may be sensitive to the detection of different types of surfaces, a robot is often equipped with multiple complementary sensors. In such cases the range measurements returned by all relevant sensors are combined into one comprehensive occupancy grid map. Bayes filters have a clear drawback in the case where sensors detect different types of obstacles since the result is ill-defined [96]. In such a situation separate maps are built for each sensor type and the maps are integrated by some combining function.

One such function is given by

$$p(m_i) = 1 - \prod_k \left(1 - p(m_i^{[k]})\right), \tag{4.5.1}$$

where $p(m_i^{[k]})$ denotes the occupancy probability of cell i as a result of the map built by sensor k. Here we assume independence between measurements of the different sensors.

If a conservative function is desired the combined map can be computed using

$$p(m_i) = \max_k p(m_i^{[k]}), \tag{4.5.2}$$

which implies that, if one map regards a cell as occupied, so does the final one.

4.6 Examples

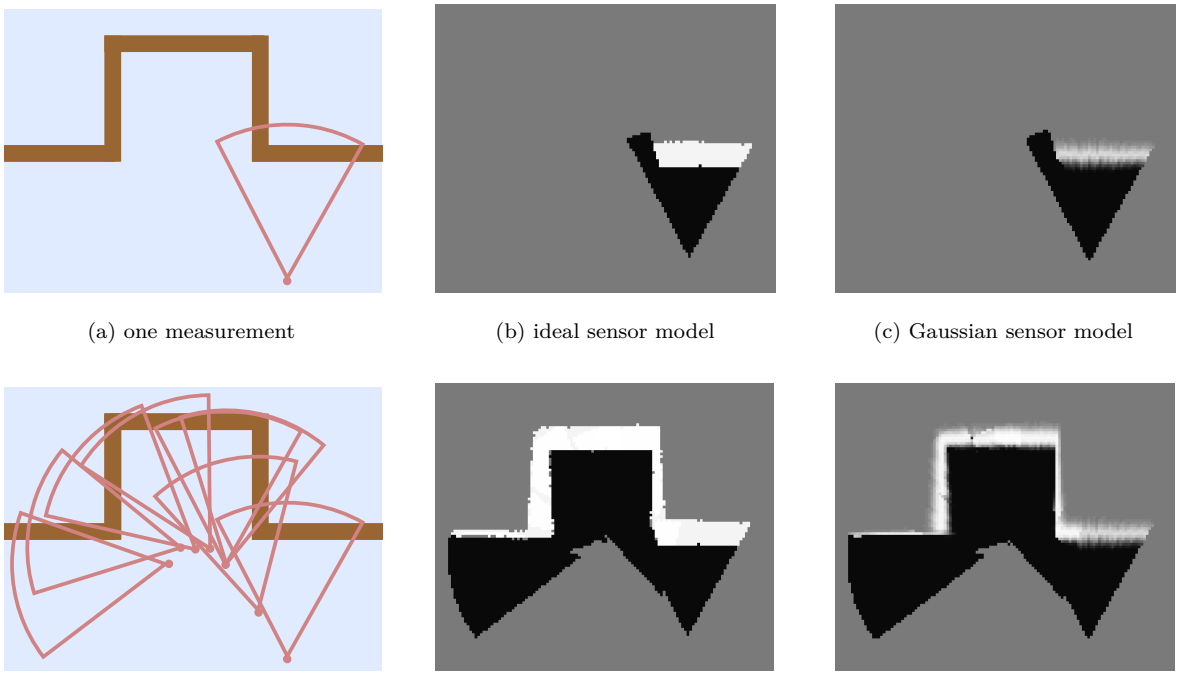
Next we provide some examples to illustrate the key concepts of this chapter.

Figure 4.8(a) shows a simulated environment and one field of view. The occupancy grid resulting from this measurement using the ideal inverse sensor model is shown in (b) and using our Gaussian inverse sensor model in (c). No measurement noise has been simulated in this example.

(a) environment and field of view (b) ideal sensor model (c) Gaussian sensor model

Figure 4.8 – Comparing the effect of the ideal sensor model and the Gaussian sensor model for a single field of view. In (a) the simulated environment is shown. The occupancy grid using the ideal sensor model is shown in (b). The occupancy grid resulting from a Gaussian inverse sensor model is shown in (c).

A case where noise is added to the input measurements is shown in Figure 4.9 for the same environment as in Figure 4.8.



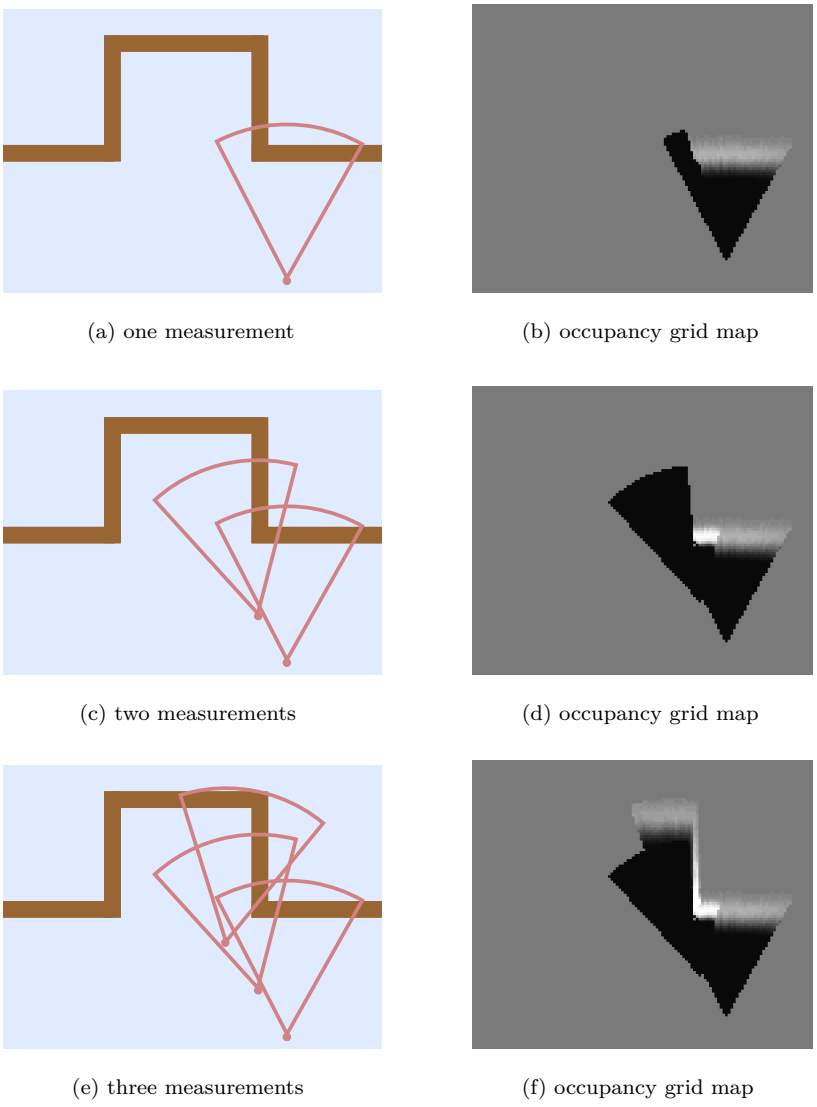
(a) one measurement (b) ideal sensor model (c) Gaussian sensor model

(d) eight measurements (e) ideal sensor model (f) Gaussian sensor model

Figure 4.9 – Comparing the effect of the ideal sensor model and the Gaussian sensor model in the presence of measurement noise.

The noise added for the example in Figure 4.9 is drawn from a uniform distribution on the interval $[-0.5, 0.5]$. The ideal sensor model assigns occupied values to an unnecessarily large region when measurement noise is present. In the case of the Gaussian sensor model the map is ostensibly more realistic. After eight consecutive measurements this effect is even more apparent, as can be seen in Figure 4.9(e) and (f). It seems that the Gaussian inverse sensor model is more robust to noise, which is as expected. For this reason we discard the ideal sensor model and employ the Gaussian inverse sensor model from here onward.

For the examples up to this point, the value of σ has been constant and not dependent on the range distance Z. In Figure 4.10 we demonstrate the effect of such a dependency. A function similar to (4.3.13) is used to generate this example.



(a) one measurement (b) occupancy grid map

(c) two measurements (d) occupancy grid map

(e) three measurements (f) occupancy grid map

Figure 4.10 – An example of the effect of a σ value, that is dependent on the measurement distance Z, on the occupancy grid map resulting from a Gaussian inverse sensor model in the presence of measurement noise.

For the first time step we see that the obstacles are relatively far from the sensor which results in a wide spread of occupied cells in (b). During the next time step the corner is located relatively close to the sensor, resulting in higher probability values and a denser concentration of occupied cells in (d) for this

region. The third measurement captures a part of the obstacle at close range and a part at a relatively large distance from the sensor. This is reflected in the occupancy grid map in (f) where the vertical part has high occupancy probability values contained in a narrow strip while the horizontal part has lower, more widely spread values.

This chapter was devoted to the incorporation of measurement uncertainty into the basic occupancy grid algorithm for a variety of sensors. In the next chapter we address the problem of incorporating pose uncertainty into the occupancy grid mapping algorithm.

Chapter 5

Pose uncertainty

The basic occupancy grid algorithm functions under the assumption that the exact pose (position and bearing of the robot) is known at every time step. In the previous chapter we incorporated measurement uncertainty into the mapping algorithm, but we know from Chapter 2 that we typically also have uncertainty in the pose estimate. We assume that, at every time step, the robot pose is available either as parameters of a closed-form pdf or as a collection of weighted samples that represent a pdf.

Efforts have been made to improve pose estimation by decreasing the associated uncertainty [40; 52]. However, the problem of incorporating pose uncertainty into the occupancy grid mapping process has received little attention. Even when information on the pose uncertainty is available, occupancy grid mapping is usually done by assuming that the most likely pose is correct [96]. The problem with this assumption is that, if the most likely pose estimate has a large associated uncertainty, the resulting map will not reflect it.

If the uncertainty in 2D pose (x, y, θ) is described by a three-dimensional Gaussian pdf (six-dimensional in 3D), as is the case in EKF SLAM, it might seem possible to simply convolve the occupancy grid map with an appropriate Gaussian kernel after the update at each time step [38]. This technique is a possibility if there is uncertainty in position only, but it becomes arduous when uncertainty in bearing must also be taken into account. Convolving the 2D map with a 3×3 covariance matrix, whose third dimension corresponds to orientation, is not that straightforward. The uncertainty in bearing is generally not negligible and, in fact, a relatively small perturbation in bearing has a large effect on the positions of distant objects in the map.

In this chapter we present a novel method to incorporate pose uncertainty into the occupancy grid algorithm. In a nutshell, we sample from the pose distribution and, for every sample, transform the measurement and add (a possibly weighted version of) it to the map using the occupancy grid update equation. If we are given a closed-form pdf, as in the case of EKF SLAM, we first need to sample from it. If we are given particles, as in the case of FastSLAM, we already have our samples and can use them directly.

5.1 Sampling via the cumulative distribution function

Our approach for incorporating pose uncertainty into occupancy grid maps requires a set of poses sampled from the pose distribution. This distribution comes from the SLAM system and may be in the form of parameters of a specific, known pdf. The EKF SLAM algorithm, for example, provides a mean vector and covariance matrix that describe a multivariate Gaussian distribution that we need to sample from. We first describe a procedure to sample from a known one-dimensional distribution and then, in the following section, expand it to the case of multivariate Gaussian distributions.

Suppose we wish to generate random samples from a known one-dimensional pdf p_x. Suppose further that we have a means to sample from the uniform distribution

$$p_u(u) = \begin{cases} 1, & \text{if } u \in [0, 1], \\ 0, & \text{otherwise.} \end{cases} \tag{5.1.1}$$

We need a transformation function of the form $x = g(u)$ so that, if $u \sim p_u$, $x \sim p_x$. Consider, for this purpose, the so-called quantile function $g(u) = F_x^{-1}(u)$, where F_x is the cumulative distribution function (cdf) of the random variable x, defined as

$$F_x(a) = p_x(x \leq a) = \int_{-\infty}^{a} p_x(x)\, dx. \tag{5.1.2}$$

In order to prove that this choice of g is suitable we must show that the cdf of the random variable $y = F_x^{-1}(u)$, where $u \sim p_u$, is equal to F_x. Indeed, since F_x is non-decreasing,

$$\begin{aligned} F_y(a) &= p_y\left(y \leq a\right) \\ &= p_y\left(F_x^{-1}(u) \leq a\right) \\ &= p_u\left(u \leq F_x(a)\right) \\ &= F_u(F_x(a)). \end{aligned} \tag{5.1.3}$$

Therefore, since u is a uniform random variable, $F_y(a) = F_x(a)$, which is what we set out to prove.

So the procedure to sample from a known one-dimensional pdf is to generate a uniform random number between 0 and 1, and compute the inverse of the cdf at that number. This procedure is termed the inverse transform method [80].

Suppose that the pdf from which we wish to sample is the one-dimensional Gaussian distribution with mean μ and variance σ^2. The cdf of this function is given as

$$F_x(x; \mu, \sigma^2) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right), \tag{5.1.4}$$

and an example is depicted as a curve in Figure 5.1.

The motivation behind using the cdf for sampling stems from the fact that the cdf has a steep slope in regions of high probability. This means that uniformly distributed F_x values result in a denser spread of x values near the mean. This concept is illustrated in Figure 5.1.

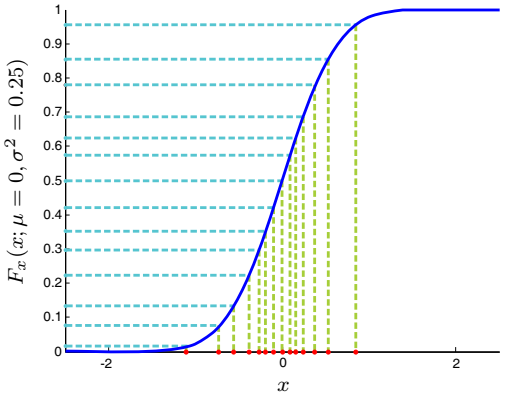


Figure 5.1 – Sampling from the Gaussian distribution with $\mu = 0$ and $\sigma^2 = 0.25$. The dots on the y-axis represent samples from the uniform distribution. Their inverses are calculated and shown as dots on the x-axis. Note that a dense cluster of x values occur close to the mean and the density decreases with distance from the mean, which is what one would expect of samples from a Gaussian distribution.

The inverse of (5.1.4) is given by

$$\begin{aligned} x &= F_x^{-1}(u) \\ &= \sqrt{2}\sigma\, \mathrm{erf}^{-1}(2u-1) + \mu, \end{aligned} \tag{5.1.5}$$

where erf^{-1} denotes the inverse of the error function stated in (4.2.10) and is computed numerically.

5.2 Decoupling multivariate Gaussian distributions

In the case of multivariate distributions the cdf is a function of multiple variables, and the inverse may be difficult to compute. A more convenient way to employ the theory in the previous section is to decouple the multivariate distribution into dimensions that can be handled separately. A random number is generated for each dimension, an inverse is computed and the multivariate sample is obtained by combining the individual samples into one vector and transforming it back to the original distribution.

A set of samples generated from a typical distribution returned by EKF SLAM is shown in Figure 5.2. The location of each sample is represented by a dot while an arrow indicates its bearing. There seems to be quite a strong relationship between the position of the sample and its bearing, implying that the variables are correlated.

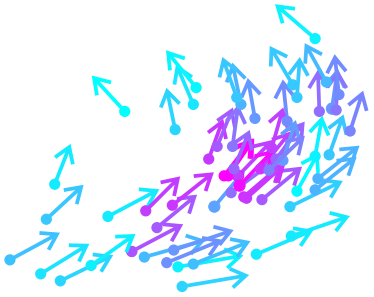


Figure 5.2 – A set of samples generated from a typical pose distribution obtained from EKF SLAM. The locations of the samples are represented by dots and their bearings by arrows. Also, the warmer the colour of the sampled pose the higher its likelihood.

Suppose the multivariate Gaussian distribution from which we wish to sample is described by a mean vector μ and covariance matrix Σ. If the variable has two dimensions, the covariance matrix is 2×2 and a confidence ellipse can be drawn around the mean. In Figure 5.3 two examples of confidence ellipses are shown. Both correspond to distributions with mean $[\,0\;0\,]^{\mathrm{T}}$. The distribution of the green ellipse has a full covariance matrix, and the blue one's covariance matrix has zero off-diagonal entries. The latter is said to be decoupled since there is no correlation between x and y. If we can find a suitable rotation to transform a full covariance into one that has zero off-diagonal entries, we can decouple the distribution, perform sampling along each dimension and rotate the obtained sample vector back to the original distribution.

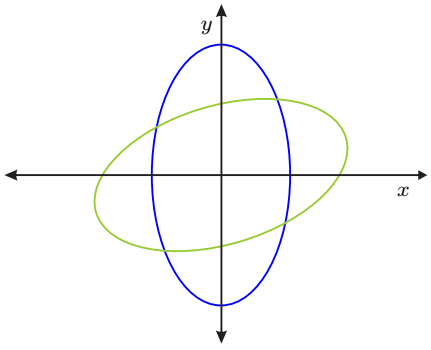


Figure 5.3 – Confidence ellipses for two 2D Gaussian distributions, both with mean $[\,0\;0\,]^{\mathrm{T}}$. The ellipse in green is generated by a distribution with a full covariance matrix and the one in blue by one where all off-diagonal entries of the covariance matrix are zero.

The covariance matrix Σ is symmetric and can be decomposed as

$$\Sigma = QDQ^{\mathrm{T}}, \tag{5.2.1}$$

where Q is an orthogonal matrix that contains the eigenvectors of Σ as columns and D is a diagonal matrix that contains the eigenvalues of Σ. It follows that

$$D = Q^{\mathrm{T}}\Sigma Q \tag{5.2.2}$$

is the covariance matrix [42] if all variables x from the original distribution are transformed to y according to

$$y = Q^{\mathrm{T}}(x - \mu)Q. \tag{5.2.3}$$

Multiplication by Q or Q^{T} can be seen as a rotation since Q is orthogonal. This rotation is the first step towards obtaining an identity covariance matrix through a process termed the whitening transform. Further steps of this transform are unnecessary in our case, and the reader is referred to [42] for more information.

Sampling is conducted by first transforming the covariance matrix Σ into the diagonal matrix D containing the eigenvalues of Σ. For each of the dimensions of the mean vector, sampling from the one-dimensional normal distribution is performed separately as in the previous section, taking the standard deviation of the ith dimension as the ith eigenvalue of Σ and the mean as the ith entry of the mean vector μ. These individual samples are combined in the same order into one sample vector y. This vector is then transformed by applying the inverse transform

$$x = QyQ^{\mathrm{T}} + \mu, \tag{5.2.4}$$

yielding the desired sample x from the original coupled multivariate distribution.

5.3 Sample size

Next we focus our attention on how many samples should be drawn. If the number of samples is too small, the result may be non-representative. On the other hand, too many samples may increase subsequent computation costs unnecessarily.

If we draw a set of M samples from a one-dimensional Gaussian distribution with mean μ and variance σ^2, the sample mean is calculated as

$$\bar{x} = \frac{1}{M}\sum_{j=1}^{M} x_j, \tag{5.3.1}$$

where x_j is the jth sample. The sample mean is also a Gaussian random variable [100] with mean μ (so it is an unbiased estimate) and variance σ^2/M, which implies that

$$\bar{x} - \mu \sim \mathcal{N}\left(0, \frac{\sigma^2}{M}\right). \tag{5.3.2}$$

If we define the sample error as the distance between the sample mean and the true mean, i.e.

$$E = |\bar{x} - \mu|, \tag{5.3.3}$$

and since one standard deviation from the mean encloses approximately 68% of the area under the Gaussian curve, we are 68% sure that

$$E \leq \frac{\sigma}{\sqrt{M}}. \tag{5.3.4}$$

To increase this confidence to roughly 95% we require that

$$E \leq \frac{2\sigma}{\sqrt{M}}. \tag{5.3.5}$$

If we decide on a maximum error $E_{\max}$, solving for M yields

$$M = \left\lceil 4 \left(\frac{\sigma}{E_{\max}} \right)^2 \right\rceil \tag{5.3.6}$$

for a confidence of at least 95%. If, for example, we set a maximum error of $E_{\max} = 0.004$ and the standard deviation is given as $\sigma = 0.015$ the number of required samples amounts to 57. Note that by taking 57 samples the sample mean is not guaranteed to be within a distance of 0.004 from the true mean, but we can be 95% sure that it will, in fact, be the case.

A multivariate distribution is factorized into multiple sampling problems (if the variables are statistically independent), one for each dimension. If we denote the jth dimension's M value by M_j, we compute the total number of samples as

$$M = \prod_{j=1}^{n} M_j, \tag{5.3.7}$$

where n is the number of dimensions. As a consequence a significantly higher number of samples must be drawn in the case of 3D pose estimates, which are described by six-dimensional distributions, compared to 2D pose estimates that are three-dimensional.

5.4 Adapted update equation

We assume that, at every time step t, the input pose distribution comes from a SLAM system. If the distribution is described by parameters of a known pdf such as a multivariate Gaussian, we draw pose samples via the inverse transform method. Since the samples are drawn from a probability distribution, the distribution of the particles describe the underlying pdf. On the other hand, in procedures such as the particle filter, the distribution of the samples alone do not reflect the pdf. Instead each sample is assigned a weight, and the weights reflect the pdf. Since we want to incorporate both representations into our algorithm equal weights are assigned if the samples are drawn from a pdf. If the pose distribution is described by a set of particles, as in the FastSLAM algorithms, we already have a set of samples and accompanying weights.

Let us therefore assume that the pose distribution at time t is described by samples

$$x_t = \left\{ x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \right\} \tag{5.4.1}$$

and corresponding weights, summing to 1,

$$w_t = \left\{ w_t^{[1]}, w_t^{[2]}, \ldots, w_t^{[M]} \right\}. \tag{5.4.2}$$

We also have a range measurement z_t given relative to the robot's pose. Using each pose sample we transform this measurement to global coordinates (as we did in Chapter 3), to obtain a set of M measurements

$$z_t = \left\{ z_t^{[1]}, z_t^{[2]}, \ldots, z_t^{[M]} \right\}, \tag{5.4.3}$$

which differ from one another only by rotation and translation.

The occupancy grid update equation from Chapter 3 is given by

$$\log \left(\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})} \right) = \log \left(\frac{p(m_i|z_t)}{p(m_i^c|z_t)} \right) + \log \left(\frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})} \right) - \log \left(\frac{p(m_i)}{p(m_i^c)} \right), \tag{5.4.4}$$

where m_i is the event that cell i is occupied. Instead of having one measurement to compute a new log odds ratio (the first term on the right hand side), we now have multiple candidates with associated weights.

The expected log odds ratio of m_i given z_t is approximated using the set of measurements z_t and their corresponding weights w_t. If v is a random variable defined as

$$v = \log \left(\frac{p(m_i|z_t)}{p(m_i^c|z_t)} \right), \tag{5.4.5}$$

the expected value is computed as

$$E[v] = \int v\, p(v)\, dv \tag{5.4.6}$$

$$\approx \sum_{j=1}^{M} w_t^{[j]} \log\left(\frac{p\left(m_i \left| z_t^{[j]}\right.\right)}{p\left(m_i^c \left| z_t^{[j]}\right.\right)}\right), \tag{5.4.7}$$

and this Monte Carlo approximation is then used as the first term in (5.4.4). The strong law of large numbers [79] ensures that this approximation will converge to the true expected value as $M \to \infty$.

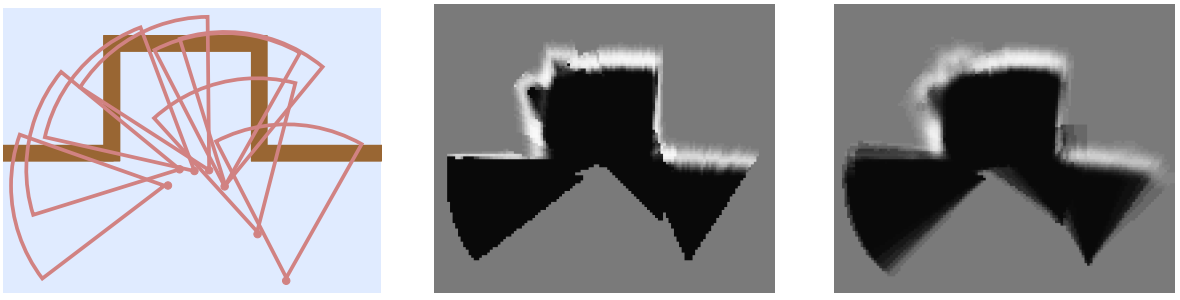
Our update equation thus becomes

$$\log\left(\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})}\right) = \sum_{j=1}^{M} w_t^{[j]} \log\left(\frac{p\left(m_i \left| z_t^{[j]}\right.\right)}{p\left(m_i^c \left| z_t^{[j]}\right.\right)}\right) + \log\left(\frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})}\right) - \log\left(\frac{p(m_i)}{p(m_i^c)}\right). \tag{5.4.8}$$

The strategy we follow to update a particular cell's value is to identify all rays for a specific sample and compute the log odds ratio as described in Chapter 4, multiply it by the sample's weight, compute the sum over all samples and add the result to the map.

5.5 Examples

Here we provide some examples to illustrate the key concepts of this chapter. In Figure 5.4 the same map used in previous examples is used, but this time pose and measurement noise is added to the system. The measurement noise is sampled from a uniform distribution on the interval $[-0.5, 0.5]$.



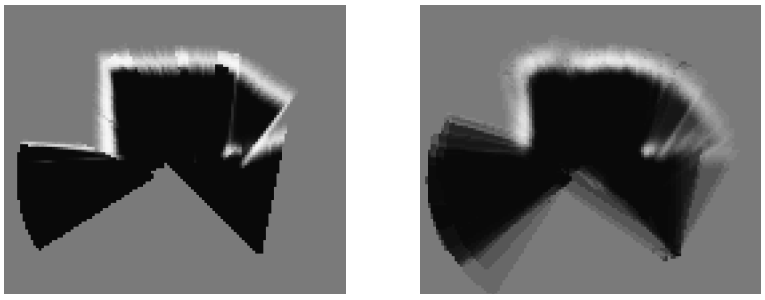
(a) eight measurements (b) without pose uncertainty (c) with pose uncertainty

Figure 5.4 – Investigating the effect of pose uncertainty on the occupancy grid map after eight measurements. In (a) the true environment and fields of view are shown. The basic occupancy grid map is shown in (b), while the occupancy grid that incorporates pose uncertainty is shown in (c).

The covariance matrix used to describe the pose uncertainty in this example is generated by choosing the first eigenvector to coincide with the bearing and by performing the Gram-Schmidt process on this and two other randomly generated vectors. Positive eigenvalues are randomly generated and ordered so that the largest eigenvalue (and thus the largest uncertainty) corresponds to the bearing eigenvector, since the greatest uncertainty typically occurs along the direction in which the robot moves. The final covariance matrix is composited from its eigenvectors and eigenvalues according to

$$\Sigma_t = Q \Lambda Q^{\mathrm{T}}, \tag{5.5.1}$$

where Q contains the eigenvectors as its columns and Λ is a diagonal matrix containing the eigenvalues. To add noise to the pose, we sample a triplet x, y and θ from the Gaussian distribution arising from the



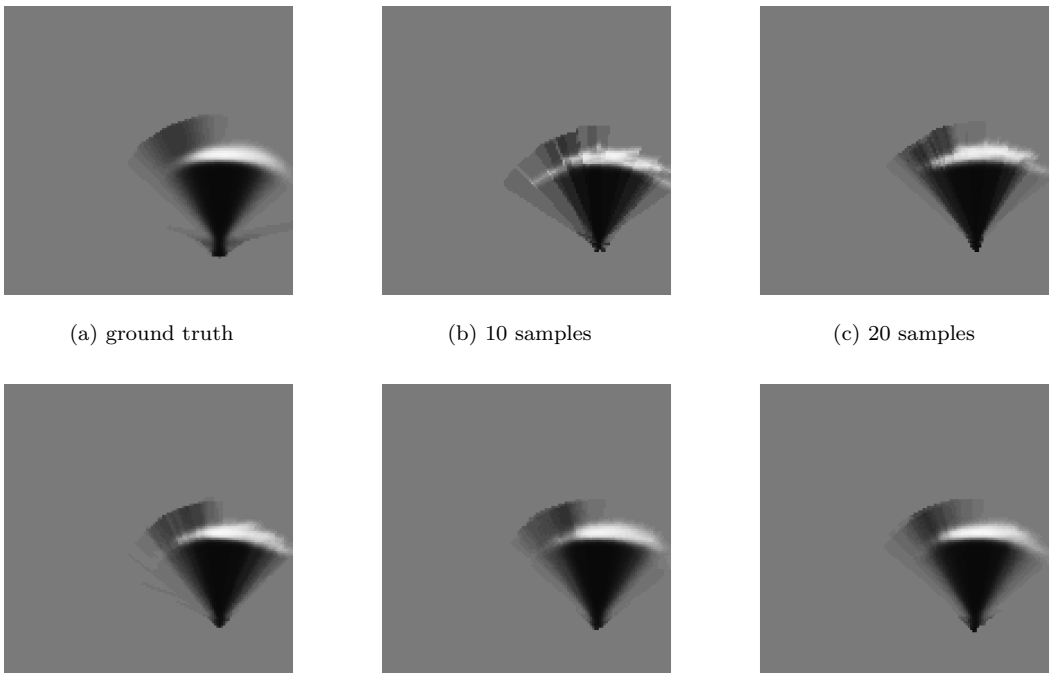
(a) without pose uncertainty (b) with pose uncertainty

Figure 5.5 – Investigating the effect of large pose uncertainty on the occupancy grid map. The basic occupancy grid map is shown in (a) and the one built with pose uncertainty is shown in (b). In this case the pose uncertainty is much greater (on average) than in Figure 5.4.

true pose and this covariance matrix. We then generate samples from the Gaussian distribution arising from the sampled pose and the covariance matrix Σ_t.

In Figure 5.4(b) the occupancy grid map resulting from assuming that the most likely pose is correct is shown. Note that several inconsistencies are present in this map. Our way of incorporating pose uncertainty combats this problem to some extent, as is shown in (c).

In Figure 5.5 the pose uncertainty is much greater. Although both maps have clear inconsistencies when compared to the true map of the environment, the effect of incorrect poses seems much less severe in the case of our proposed algorithm. For example, the measurement of the top right hand corner had a large error in bearing, but because the uncertainty is so great, this measurement is blurred over a large region. The measurement of the top left hand corner seems to have had a relatively small uncertainty since this measurement is hardly blurred at all.



(a) ground truth (b) 10 samples (c) 20 samples

(d) 60 samples (e) 120 samples (f) 180 samples

Figure 5.6 – Investigating the effect of sample size. A ground truth map built by using 10^5 samples is shown in (a). In (b)–(f) the maps built with different sample sizes are shown.

Next we investigate the effect that the sample size has on the resulting map. For this experiment we generate a random covariance matrix for the first field of view in Figure 5.4(a) and build a map using 10^5 samples, which we regard as a ground truth map that represents the associated uncertainty. This map is shown in Figure 5.6(a) and (b)–(f) show the map built from 10, 20, 60, 120 and 180 samples. We see that, when too few samples are used, differences between those maps and the ground truth are easily identifiable. However, as the number of samples increase, the information gained becomes significantly less.

In this chapter we discussed a means to handle pose uncertainty in the context of occupancy grids. In the next chapter we address another shortcoming of the basic occupancy grid algorithm, namely its demanding memory requirements.

Chapter 6

Adaptive occupancy grid mapping

When designing an occupancy grid mapping algorithm, the choice of resolution (i.e. the grid size) can be extremely important. If a grid cell is too large it can easily be partially occupied by some obstacle but, since only one occupancy probability is stored and updated per cell, inconsistencies may arise. On the other hand, if the grid is too fine, unnecessary computation is conducted if, for example, a large region over many cells is completely occupied or completely unoccupied. As the environment may have large homogeneous regions as well as regions with more variation in occupancy, an algorithm that can adapt its grid size depending on the measurements it receives would be advantageous for limiting inconsistencies while remaining efficient.

The result of an algorithm that can adapt its grid size locally is an array of squares or cubes with different side lengths. One way of representing such a map would be to use a spatial data structure such as a region tree. The idea of representing a map as a region tree has received some attention [59; 76; 105], but mostly in the post-processing paradigm after an initial map has been constructed.

An efficient 3D mapping algorithm has been proposed by Wurm et al. [104], which uses region trees as part of an online algorithm. Measurements are segmented into regions and each region is updated separately in a hierarchical tree, which results in multiple resolutions. However, this approach requires user input for the segmentation process.

In this chapter we discuss a fully automatic mapping algorithm that adapts its grid size on-the-fly. Our approach is based on a method by Einhorn et al. [37].

6.1 Spatial data structures

As mentioned, spatial data structures have been used to represent occupancy grid maps. In this section we introduce spatial data structures in the form of region trees and explain their function in the occupancy grid framework.

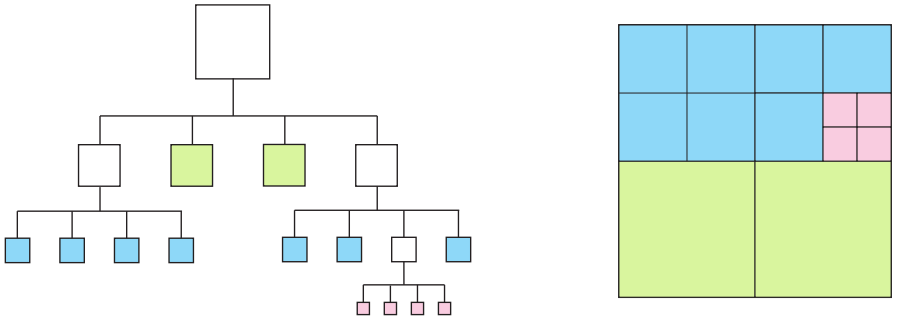
6.1.1 Definition

A region tree consists of nodes connected in a tree structure. In 2D every node corresponds to a square in the plane, described by the coordinates of its centre and its side length. As before, the squares are axis aligned so that these three parameters are sufficient.

We start with a single root node at the top of the tree structure. The square associated with this root node can be halved along each of its dimensions, resulting in four children with side lengths half that of their parent. Each one of these four children can be divided again, leading to four more nodes for each, and this process ultimately results in $4^{\ell-1}$ nodes on the ℓth level. The procedure of dividing nodes continues until a prespecified number of levels is reached which, in effect, sets the maximum resolution of the grid.

These tree structures are commonly known as region quadtrees or 2^2-trees. They form part of a more general class of trees called N^d-trees, specifying that a cell in d dimensions is divided into N parts along each dimension [30]. The 3D analogue of the quadtree is therefore the 2^3-tree or the octree. In this work we consider quadtrees for the 2D case and octrees for the 3D case.

An irregular spatial grid structure is obtained by allowing nodes on different levels to be active, by which we mean we have removed all its children (if it had any). These are appropriately named leaf nodes. Figure 6.1(a) depicts a quadtree where the leaf nodes are indicated in colour and the corresponding grid can be seen in (b). Note that the area covered by all the leaf nodes equals the area of the root node and no leaf node regions overlap. This restriction can be enforced by choosing either a parent or its four children, i.e. allowing neither a partial parent nor a subset of siblings to appear in the map.



(a) quadtree (b) corresponding grid

Figure 6.1 – A graphical representation of a quadtree. In (a) the tree structure is portrayed. Note that there are different levels, each corresponding to a grid cell of a particular size. The leaf nodes (nodes with no children) may be on different levels and are coloured accordingly. In (b) the corresponding spatial grid structure is shown using the same colours.

Since grid cells in 3D has a width, depth and height, bisecting each of the dimensions results in $2^3 = 8$ children for each node. This means that on the ℓth level there are $8^{\ell-1}$ nodes in total, as opposed to $4^{\ell-1}$ nodes for a quadtree. A simple representation of this hierarchy is shown in Figure 6.2.

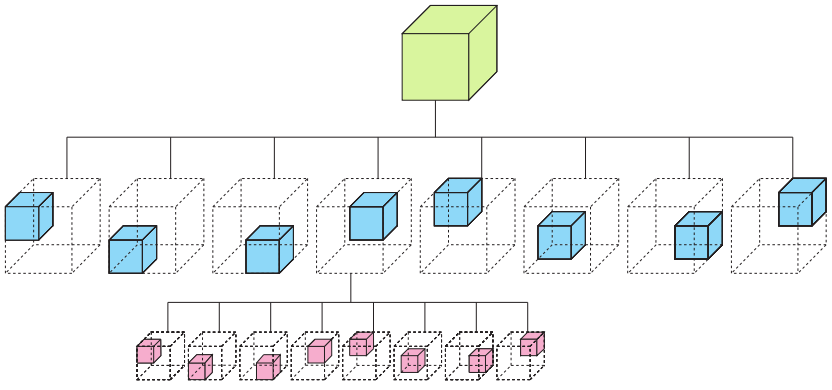


Figure 6.2 – A schematic representation of a region octree. A cell can be split into eight smaller ones by bisecting the cell along each of its dimensions.

6.1.2 Application to occupancy grids

A predefined area in the 2D plane (or 3D space) can be modelled by means of a quadtree (or octree). In the occupancy grid paradigm this would mean that, instead of having a regularly spaced lattice of cells, we now have a tree structure with leaf nodes describing cells of different sizes that cover the same area.

Suppose we have a root node representing the total area to be mapped. We initialize all leaf nodes on a prespecified coarsest level. Now, if we receive our measurement at time t, we perform tests on each leaf node in the field of view to determine if this node should be split into four children. If so, it is split recursively until the resulting leaf nodes no longer satisfy the split criteria or they reach a prespecified finest level. Once all splits are done, leaf nodes are assigned probability values according to the update equation of the occupancy grid algorithm. The algorithm then checks for sets of siblings that are eligible for merging, and performs the merge if the nodes satisfy certain criteria, which can also be a recursive process. Details of these tests and criteria for the splitting and merging of nodes are discussed next.

6.2 Splitting a node

A node should be split if its current resolution is inadequate to model the perceived environment as observed by the sensor. In other words, we would like to split a node if it receives conflicting information regarding its occupancy. This means that all measurement rays that intersect a specific node should be identified. A ray traversal algorithm designed especially for region trees can be used, such as the one in [1]. Once all such rays are identified, and if the leaf node is not on the finest level allowed, we have to decide whether or not to split the node. What further complicates this decision is that conflicts may arise as a result of measurement noise and not only from the true state of the environment.

6.2.1 Counting hits and misses

In order to make our decision, we keep track of the number of hits and misses of the cell, denoted by n_o and n_f respectively. The output from the sensor returns the distance to the first observed obstacle along rays of sight. If no obstacle is observed along a specific ray, the maximum range measurement is returned.

To determine the number of hits and misses for a cell, we identify all rays from the sensor that would, in the absence of obstacles, pass through the cell. The number of hits corresponds to the number of these rays indicating an obstacle inside the boundaries of the cell. The number of misses corresponds to the number of rays passing through the cell, either because the first obstacle falls behind the cell or the maximum range is returned. Note that hits or misses are the only two possibilities considered, implying that the number of hits added to the number of misses should equal the number of rays passing through the cell. If an obstacle is observed before the ray reaches the cell, the contribution of that ray and its measurement are ignored for that particular cell.

In the previous chapter we incorporated pose uncertainty into the mapping algorithm by sampling from the pose uncertainty distribution. We also mentioned that at a particular time we receive multiple samples and thus multiple measurements. We identify all rays that pass through the cell across all the samples and so determine the total number of hits and misses, making no distinction between multiple rays from a single measurement and multiple rays from different measurements.

If a cell is partially occupied and observed by multiple rays from a noise-free sensor, some rays may return hits and some misses. This is an example of the case where a cell should be split because the current resolution is inadequate. On the other hand, if there are only hits and no misses, the entire cell is occupied. Similarly, if only misses and no hits are observed, the entire cell is free. This also holds for measurements over time, but since an incremental algorithm is proposed we consider only the current number of hits and misses when deciding to split. However, these number counts are propagated to the next time step so that if a new measurement proves the current (adequate) resolution to be inadequate, the cell may still be split even if the new measurement has no mutually contradicting evidence.

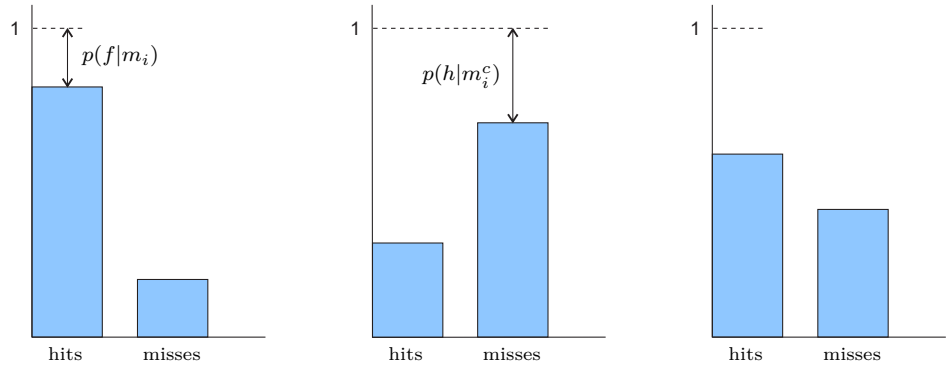
In practice we must deal with noisy measurements. Following the approach in [37], we denote the probability that the sensor returns a hit if cell i is in fact free by $p(h|m_i^c)$ and the probability that it returns a miss if a particular cell is in fact occupied by $p(f|m_i)$. Here we assume independence between measurement rays for both of these probabilities. These quantities can be determined experimentally for a specific sensor by, for example, placing obstacles at known distances in the field of view of the sensor and calculating the detection frequencies.

If M measurement rays pass through a cell the expected number of hits and misses, depending on the state of the cell, is given by

$$E_{n_o,m_i} = M\left(1 - p(f|m_i)\right), \tag{6.2.1}$$
$$E_{n_f,m_i} = Mp(f|m_i), \tag{6.2.2}$$
$$E_{n_o,m_i^c} = Mp(h|m_i^c), \tag{6.2.3}$$
$$E_{n_f,m_i^c} = M\left(1 - p(h|m_i^c)\right). \tag{6.2.4}$$

Here the first subscript of E denotes either the number of hits (n_o) or misses (n_f) and the second denotes the true state of the cell, which can be either occupied (m_i) or free (m_i^c).

Normalized histograms, as shown in Figure 6.3, depict these expected observations. In (a) the cell is in fact occupied and a noisy sensor returns many more hits than misses. In (b) the cell is free and many more misses than hits are returned. The histogram in (c) differs significantly from (a) and (b) and its values cannot be explained by the measurement noise. This is an example of where the resolution is inadequate and the cell should be split.



(a) occupied cell (b) free cell (c) partially occupied cell

Figure 6.3 – Comparing the measured number of hits and misses to normalized histograms of the expected number of hits and misses from a noisy sensor. In (a) the cell is occupied and the sensor returns many more hits than misses. In (b) the cell is free and the sensor returns many more misses than hits. The measured number of hits and misses in (c) for a specific cell differs significantly from (a) and (b) and therefore cannot be explained solely by sensor noise. It thus represents an inadequate grid cell resolution as it may be partially occupied.

The task at hand is therefore to identify distributions such as the one in Figure 6.3(c) which differ significantly from the expected distributions in (a) and (b). This can be done by employing the chi-squared test (χ^2-test) commonly used in hypothesis testing [100].

6.2.2 The chi-squared test

In order to compare a particular distribution to multiple other distributions, the χ^2-test is performed by calculating the quantity X^2 according to

$$X^2 = \sum_{j=1}^{K} \frac{(O_j - E_j)^2}{E_j}, \tag{6.2.5}$$

where E_j is the expected number of occurrences of the jth variable, O_j is the actual observed number of occurrences and K is the number of variables. The probability distribution that applies to the statistical

test in (6.2.5) is the χ^2 distribution [100] with pdf

$$f(x;v) = \begin{cases} \dfrac{x^{(v-2)/2}e^{-x/2}}{2^{v/2}\Gamma\left(\frac{v}{2}\right)}, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6.2.6}$$

Here $\Gamma(\,\cdot\,)$ represents the gamma function and $v = K - 1$ the number of degrees of freedom. In our case $v = 1$, since the observable variables are hits and misses. This distribution function is portrayed in Figure 6.4.

A critical value corresponding to a statistical significance α can now be calculated. The shaded region in Figure 6.4 to the right of this x-value represents a fraction equal to $1 - \alpha$ of the total area under the curve. We denote this value by $\chi^2_{1,1-\alpha}$, and find it either from a table [85] or numerically. This critical value will be used in the final decision to split a node.

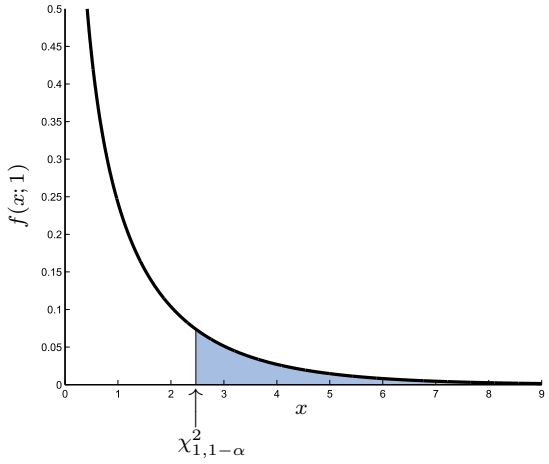


Figure 6.4 – The χ^2 distribution given in (6.2.6) for $v = 1$. A critical value $\chi^2_{1,1-\alpha}$ is indicated on the x-axis. This value is such that the area under the curve to the right of it is equal to a fraction $1 - \alpha$ of the total area under the curve.

If an X^2 value is calculated according to (6.2.5) and found to be larger than the critical value, the hypothesis that this distribution is generated by the assumed state is rejected. We may, for example, choose a statistical significance of $\alpha = 0.995$. This implies that if an X^2 value is less than the corresponding critical value of about 7.897, we are 99.5% sure that the distribution is generated by the assumed state.

In our case there are two observable quantities, n_o and n_f, and two possible states that a cell can assume, so that equation (6.2.5) becomes

$$X^2_{m_i} = \frac{\left(n_o - E_{n_o,m_i}\right)^2}{E_{n_o,m_i}} + \frac{\left(n_f - E_{n_f,m_i}\right)^2}{E_{n_f,m_i}}, \tag{6.2.7}$$

$$X^2_{m_i^c} = \frac{\left(n_o - E_{n_o,m_i^c}\right)^2}{E_{n_o,m_i^c}} + \frac{\left(n_f - E_{n_f,m_i^c}\right)^2}{E_{n_f,m_i^c}}. \tag{6.2.8}$$

A node should then be split if the observed hits and misses resemble neither the distribution corresponding to a completely occupied cell nor the one corresponding to a completely free cell. This implies that we split a cell if the minimum of $X^2_{m_i}$ and $X^2_{m_i^c}$ is larger than the specified critical value, i.e. if

$$\min\left(X^2_{m_i}, X^2_{m_i^c}\right) > \chi^2_{1,1-\alpha}. \tag{6.2.9}$$

6.2.3 Exceptions

Performing the test above on its own may not be sufficient and some exceptions can occur. If a sensor behaves better than expected, e.g. by returning all hits and no misses if the cell is occupied, the test above may identify this situation as a deviation from the expected distributions. However, this is not a case where the node should be split. Therefore, if perfect measurements such as these are observed, we do not perform the test in (6.2.9) and do not split the cell.

Another problem may arise as a result of the fact that the inverse sensor model is evaluated at a single point that represents the occupancy of the entire cell. If for example the inverse sensor model returns a value corresponding to free and the number of hits is nonzero, we split the cell regardless of the outcome in (6.2.9).

In addition, we found that by implementing the algorithm described in [37] directly, some anomalies may arise. Firstly, we split cells that are only partially in the field of view, as we cannot assume that the unobserved part has the same state as the observed part. Indeed, we would like to update only the observed part and this necessitates a node split.

Secondly, consider the scenario depicted in Figure 6.5. The area shaded blue is free, as observed by the noise-free sensor, while all grey areas are unobserved. The brown region indicates an obstacle seen by the sensor and the darker grey areas are obstacles not seen by the sensor at this particular time. The dashed line indicates the boundaries of a specific grid cell (the other cells are omitted for clarity). Measurement rays such as z_1 will have no effect on this grid cell as it will not result in a hit or a miss. On the other hand, rays such as z_2 will return misses and no hits, which will imply that this cell will not be split. This cell will then receive a probability value indicating that the entire cell is free, even though it was only partially observed and may not be completely free.

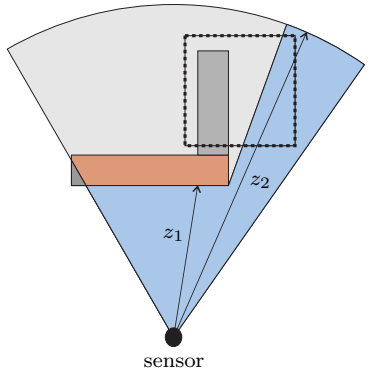


Figure 6.5 – An example of the case where the original adaptive grid mapping algorithm introduces inconsistencies. The darker grey areas are unseen obstacles while the brown region indicates an observed obstacle. The grid cell indicated by the dashed line will not be split, since measurements such as z_1 have no effect on it and measurements such as z_2 return only misses and no hits, although the cell is only partially observed and may in fact be partly occupied.

In an effort to combat this problem, we augment the algorithm described in [37] by also keeping track of the number of unknowns a cell receives.

6.2.4 Incorporating the number of unknowns

In Figure 6.5 we saw that the algorithm as proposed by [37] may introduce inconsistencies, even in the noise-free case. We propose augmenting the algorithm by adding the number of unknowns to the list of observables in the χ^2-test. We regard a measurement as an unknown if a hit occurs along the ray before it reaches the evaluated cell.

To use the χ^2-test we have to build distributions for the cases in which we do not want to split the node. We then perform the test to identify observations that differ significantly from these distributions. As before, we do not want to split the node if it is entirely occupied or entirely free. If a cell is entirely unknown, i.e. unseen by the sensor, we do not want to split it and we avoid assigning a new probability value to this cell. For this reason, we would still like to detect this situation. All other cases, i.e. partially occupied or partially unseen, should be split.

Let us first consider the case where the cell is entirely occupied. Since we include unknowns as one of the possible outcomes, we now require that

$$p(u|m_i) = 1 - p(h|m_i) - p(f|m_i), \tag{6.2.10}$$

where u denotes the event of observing an unknown. This probability may be hard to calculate experimentally, but can be set to a small value since, if a cell is observable and occupied, it is unlikely that a ray will return an unknown.

The corresponding expected number of hits, misses and unknowns for M independent measurement rays that pass through the cell at a particular time are now given by

$$E_{n_o,m_i} = Mp(h|m_i), \tag{6.2.11}$$
$$E_{n_f,m_i} = Mp(f|m_i), \tag{6.2.12}$$
$$E_{n_u,m_i} = Mp(u|m_i), \tag{6.2.13}$$

where n_u is the number of observed unknowns. The quantity $X^2_{m_i}$ now becomes

$$X^2_{m_i} = \frac{(n_o - E_{n_o,m_i})^2}{E_{n_o,m_i}} + \frac{\left(n_f - E_{n_f,m_i}\right)^2}{E_{n_f,m_i}} + \frac{(n_u - E_{n_u,m_i})^2}{E_{n_u,m_i}}. \tag{6.2.14}$$

Similarly, in the case where the cell is completely free, we require that

$$p(u|m_i^c) = 1 - p(h|m_i^c) - p(f|m_i^c), \tag{6.2.15}$$

and the expected number of hits, misses and unknowns are

$$E_{n_o,m_i^c} = Mp(h|m_i^c), \tag{6.2.16}$$
$$E_{n_f,m_i^c} = Mp(f|m_i^c), \tag{6.2.17}$$
$$E_{n_u,m_i^c} = Mp(u|m_i^c). \tag{6.2.18}$$

The probability $p(u|m_i^c)$ is also hard to calculate but, as before, it may be set to a relatively small value since an observable free cell is unlikely to return an unknown. The quantity $X^2_{m_i^c}$ now becomes

$$X^2_{m_i^c} = \frac{\left(n_o - E_{n_o,m_i^c}\right)^2}{E_{n_o,m_i^c}} + \frac{\left(n_f - E_{n_f,m_i^c}\right)^2}{E_{n_f,m_i^c}} + \frac{\left(n_u - E_{n_u,m_i^c}\right)^2}{E_{n_u,m_i^c}}. \tag{6.2.19}$$

Since we now have three observables (hits, misses and unknowns), the number of degrees of freedom in (6.2.6) is $v = 2$.

This means that we can no longer test the outcome against the critical value $\chi^2_{1,1-\alpha}$, but have to substitute it with $\chi^2_{2,1-\alpha}$, which in our case corresponds to 10.597 for $\alpha = 0.995$. So, as in (6.2.9), we do not split if

$$\min\left(X^2_{m_i}, X^2_{m_i^c}\right) \leq \chi^2_{2,1-\alpha}, \tag{6.2.20}$$

but if the converse situation occurs we still have to test if the state of the cell can be regarded as unknown. Also, if $n_f = 0$, $n_o = M$ and $n_u = 0$, the sensor performs better than expected and should be handled separately, as discussed in section 6.2.3. The same applies when $n_f = M$, $n_o = 0$ and $n_u = 0$.

The third case we are interested in is the entirely unknown case, by which we mean that the cell is not visible and therefore the state is unknown. If measurement errors are likely, this case becomes important to detect because if, for example, we have very few hits and many unknowns, it does not seem feasible

to split the cell since virtually no information regarding its state is available. Also, we do not want to assign the probability value associated with these hits to the cell, as they may be wrong. In this case we would prefer not to assign a probability value to the cell.

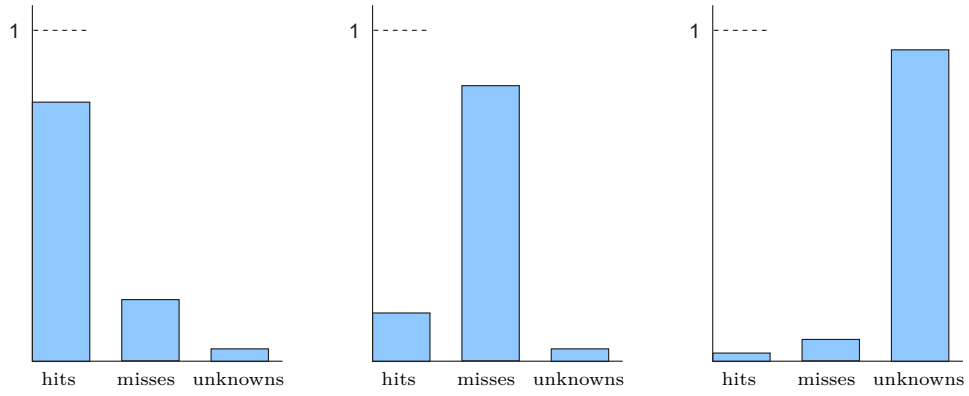
To build this distribution, we require that

$$p(u|m_i^u) = 1 - p(h|m_i^u) - p(f|m_i^u), \tag{6.2.21}$$

where m_i^u indicates that cell i is unknown. This probability may also be hard to calculate, but can be set to a large value, for example larger than 0.8, since if the cell is truly unobservable a ray will probably return an unknown. The expected number of occurrences for this distribution is given by

$$E_{n_o,m_i^u} = Mp(h|m_i^u), \tag{6.2.22}$$
$$E_{n_f,m_i^u} = Mp(f|m_i^u), \tag{6.2.23}$$
$$E_{n_u,m_i^u} = Mp(u|m_i^u). \tag{6.2.24}$$

An example of each of the three distributions discussed in this section can be seen as a normalized histogram in Figure 6.6. In (a) the cell is completely occupied and therefore receives many hits, few misses and few unknown counts. The histogram in (b) is generated by a completely free cell and registers few hits, few unknowns and many misses. The cell in (c) is completely obscured from view and receives many unknowns, few hits and few misses.



(a) occupied cell (b) free cell (c) unknown

Figure 6.6 – Normalized histograms for the three distributions arising when unknown is added to the algorithm in [37] as a possible state. The case where the cell is entirely occupied is shown in (a), the case where it is entirely free in (b) and the case where the cell is not visible by the sensor and the state is unknown is shown in (c).

There are now three distributions to which an incoming set of hits, misses and unknowns should be compared. The third X^2 value is given by

$$X^2_{m_i^u} = \frac{\left(n_o - E_{n_o,m_i^u}\right)^2}{E_{n_o,m_i^u}} + \frac{\left(n_f - E_{n_f,m_i^u}\right)^2}{E_{n_f,m_i^u}} + \frac{\left(n_u - E_{n_u,m_i^u}\right)^2}{E_{n_u,m_i^u}}, \tag{6.2.25}$$

and we regard a set of observations to be generated by the completely unknown state, and do not assign a probability value, if

$$X^2_{m_i^u} \leq \chi^2_{2,1-\alpha}. \tag{6.2.26}$$

If neither (6.2.20) nor (6.2.26) is satisfied, the node is split.

For the entirely unknown case there also arises an exception if $n_f = 0$, $n_o = 0$ and $n_u = M$, which is tested separately and handled as if it belongs to the unknown state distribution.

It was mentioned in section 6.2 that the hit and miss counts are propagated over time. The unknown count, however, is not propagated. If in one view a cell receives numerous unknown values, they may all be ignored if the cell is visible in a later view.

Now that details on all the tests to determine whether or not a node should be split (which may be a recursive process) are in place, we proceed to the updating of a cell's probability value.

6.2.5 Assigning probability values

Once a node is split and its children activated, these nodes have to receive new probability values. For the sake of convenience, we restate the basic occupancy grid update equation for cell i at time t:

$$l_{i,1:t} = l_{i,t} + l_{i,1:t-1} - l_0, \tag{6.2.27}$$

which consists of the new log odds ratio (first term) added to all previous log odds ratios (second term) minus the prior.

The new log odds ratio for the children nodes can be calculated easily by using the inverse sensor model once the relevant measurements have been identified. Although these nodes did not exist at the previous time step, their parent has stored information from the previous time steps. The children inherit this previous probability value since they occupy the same space as their parent.

6.3 Merging nodes

Once all necessary splits have been done and each leaf node's probability value has been updated according to (6.2.27), we check to see if some cells can be merged or pruned from the tree.

6.3.1 Criteria

In order to preserve the tree structure, we will only allow four leaf nodes with the same parent to be merged into that parent. Merging is done provided all four children satisfy certain criteria, similar to the algorithm in [37].

Four children in a quadtree (eight for an octree) with a common parent are candidates for merging if they are not on the predefined coarsest level. Merging may then take place if they meet at least one of the following requirements:

- the standard deviation of their current probability values is less than some predefined threshold and their mean is within some threshold distance to 0 (certainly free) or 1 (certainly occupied);
- all probability values are less than a predefined value p_{m^c};
- all probability values are greater than a predefined value p_m.

This ensures that cells with a high probability of being occupied or free are merged together, which in a way minimizes the loss of information since it is unlikely that these probabilities will be significantly altered by future measurements. These requirements also prevent the merging of cells that are partially occupied. Once four children are merged into their parent the process is repeated recursively for the parent and its siblings, as long as the criteria are met.

6.3.2 Assigning probability values

Once children have been merged into their parent, the parent has to receive a probability value. We decide to assign the mean of the children's probability values to the parent. Mathematically this is sound because the current probability values are in the form of log odds ratios. By computing the mean, i.e. adding all log odds ratios and dividing by four, we still have a log odds ratio that can be converted to a probability value as in Chapter 3. Furthermore, little information is lost since we merge only siblings that have almost the same values or are almost certainly occupied or free.

6.4 Adapting the total map size

In the original occupancy grid framework the total map size is predefined. However, by representing the map as a tree, we can alter the total map size adaptively.

For the first measurement, we start by choosing a root node that covers the entire field of view and initialize all nodes on a prespecified coarsest level. This coarsest level is typically finer than the root node since the first split test will most likely cause this node to be split anyway. We then proceed to split and merge nodes and also update the probability values of the leaf nodes.

In the next time step the robot may move to a different location that is not fully covered by the current tree. In this case, a new root node on the same level as the first one is added so that their boundaries align. Root nodes are added in this fashion until the entire field of view of the current measurement is covered by the tree, and the adaptive grid mapping algorithm is resumed. Figure 6.7 illustrates this map growing process.

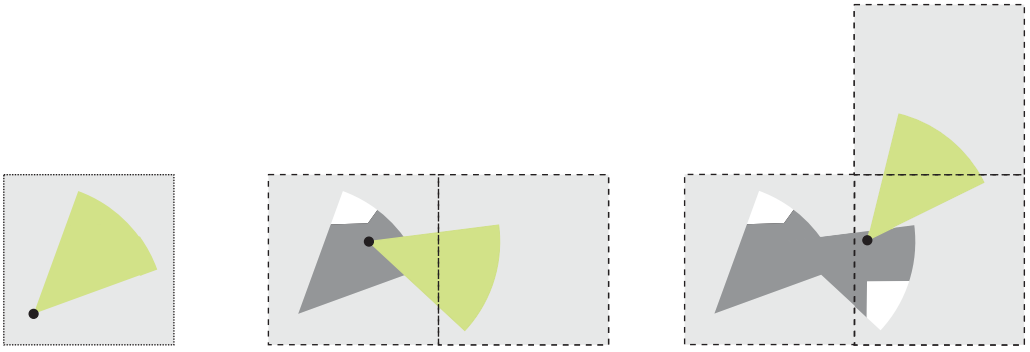


Figure 6.7 – Adaptively growing the mapped region by adding root nodes to incorporate the current field of view. The current field of view is shown in green, unknown areas in light grey, free space in dark grey and obstacles in white. The current sensor position is indicated by the black dot and only root nodes are shown for clarity.

If the purpose of mapping is purely for immediate navigation, or if memory is limited, we may discard root nodes if they have not been seen for quite some time. If the robot is expected to return to some area, however, newly obtained regions have to be compared to regions that were visible a long time ago, implying that all information should be stored.

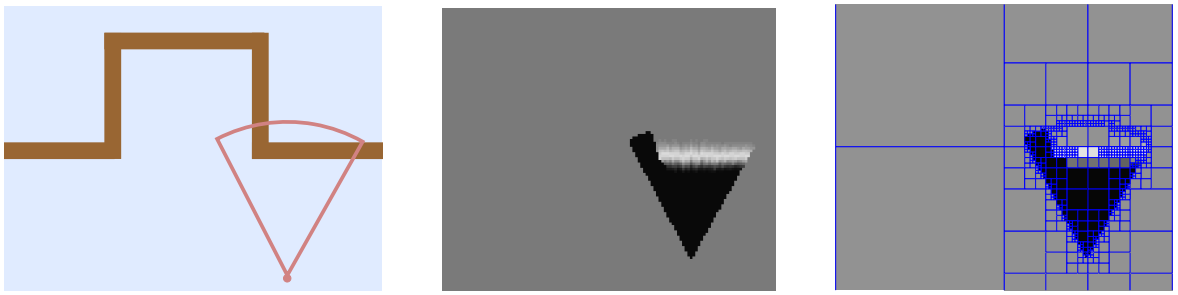
6.5 Examples

To illustrate the working of the techniques explained in this chapter, we present some examples. We first consider the case where the exact pose is known.

Figure 6.8(a) shows a field of view in a simulated environment. The occupancy grid where all cells are the same size is shown in (b). In (c) the adaptive occupancy grid is shown, where we kept track of the hits and misses only. The blue line segments indicate the boundaries of the grid cells. The finest allowed level of the quadtree is the same resolution as the occupancy grid map in (b).

In Figure 6.8(c) we see that regions that have a high probability of being free are represented by larger blocks than those regions close to the obstacles, which is as desired. We also note that in front of the horizontal wall a few blocks have a slightly larger size than those behind them and are coloured grey, which does not correspond exactly to the occupancy grid map in (b). This is because these cells receive only misses, and no hits, so there is no reason to split them. These cells each receive a single probability value according to the inverse sensor model and, since they lie between free space and the obstacle, they receive a value of close to unknown ($p(m_i|z_t) = 0.5$).

Figure 6.9 shows the maps obtained after adding two more measurements. Here the adaptive occupancy grid mapping algorithm makes a significant error. The cells circled in red in Figure 6.9(b) receive

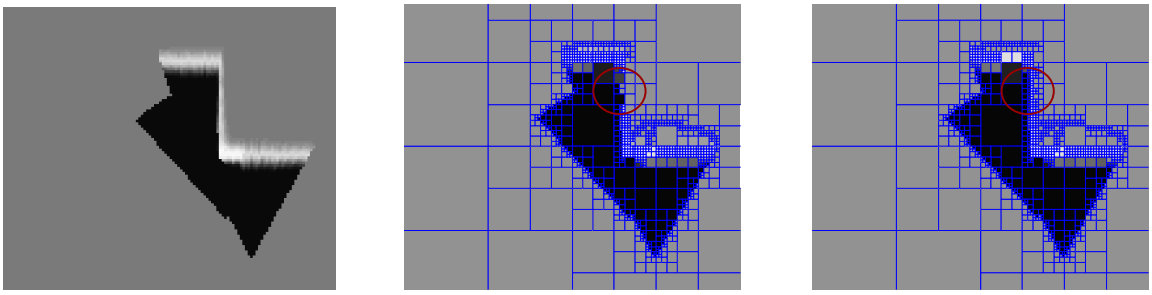


(a) environment and field of view (b) regular grid (c) adaptive grid

Figure 6.8 – Comparing the adaptive occupancy grid map to its regular counterpart for one field of view.

probability values indicating that they are free, when in reality they are not. This is an example of where the original algorithm proposed in [37] introduces inconsistencies, as we discussed in section 6.2.3.

When the same measurements are used in our adaptive mapping algorithm that also keeps track of the number of unknowns, that error disappears. This can be seen in Figure 6.9(c) where the circled cells now indicate occupancy, contrary to the error in (b).



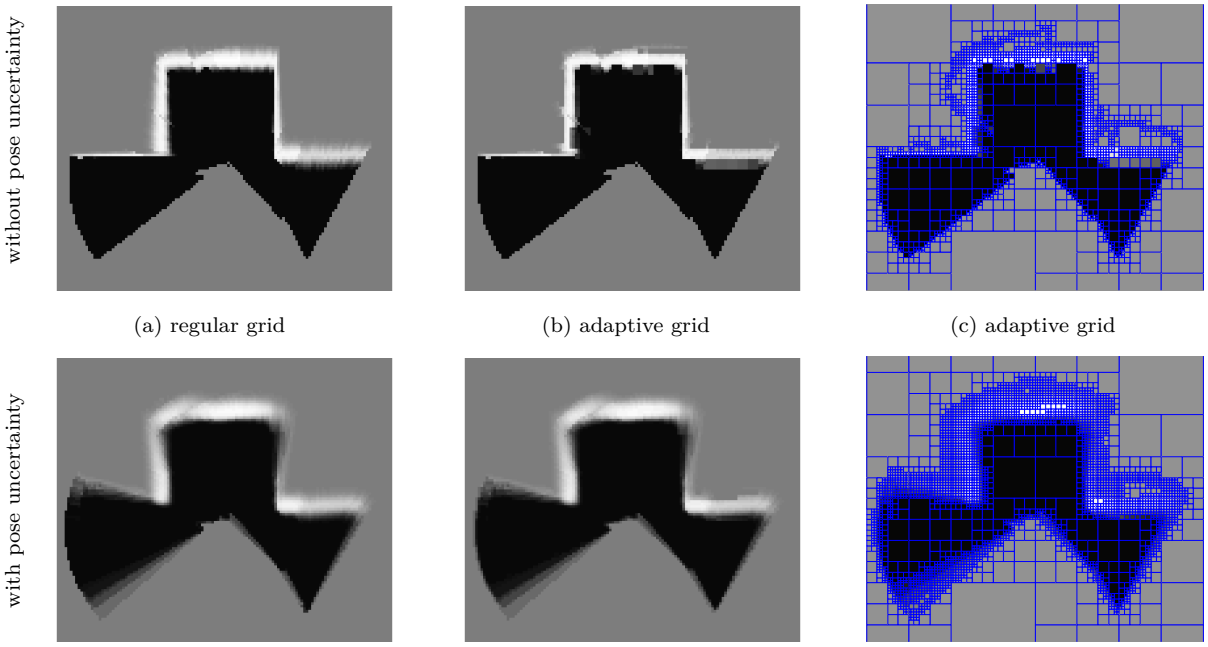
(a) regular grid (b) without counting unknowns (c) with counting unknowns

Figure 6.9 – Comparing the adaptive occupancy grid without counting unknowns with the one that does keep track of unknowns, after three measurements (the environment and fields of view are shown in Figure 4.10(e)). Note that in (c) the error that occurs in (b) is corrected.

The map from our adaptive algorithm that keeps track of unknowns, but without incorporating pose uncertainty, after eight measurements can be seen in Figure 6.10(b)–(c), with the regular counterpart in (a). Here we see that the regular and adaptive maps are virtually the same. However, the regular grid map has roughly 16 000 cells, while the adaptive map has only about 2 000.

If we investigate the incorporation of pose uncertainty, we see that the adaptive algorithm with unknowns performs quite well. In Figure 6.10(f) we see that the adaptive algorithm now requires more cells on the finest level than in (c), since more cells receive conflicting information. In this case the adaptive grid has roughly 3 800 cells, which is still significantly less than 16 000.

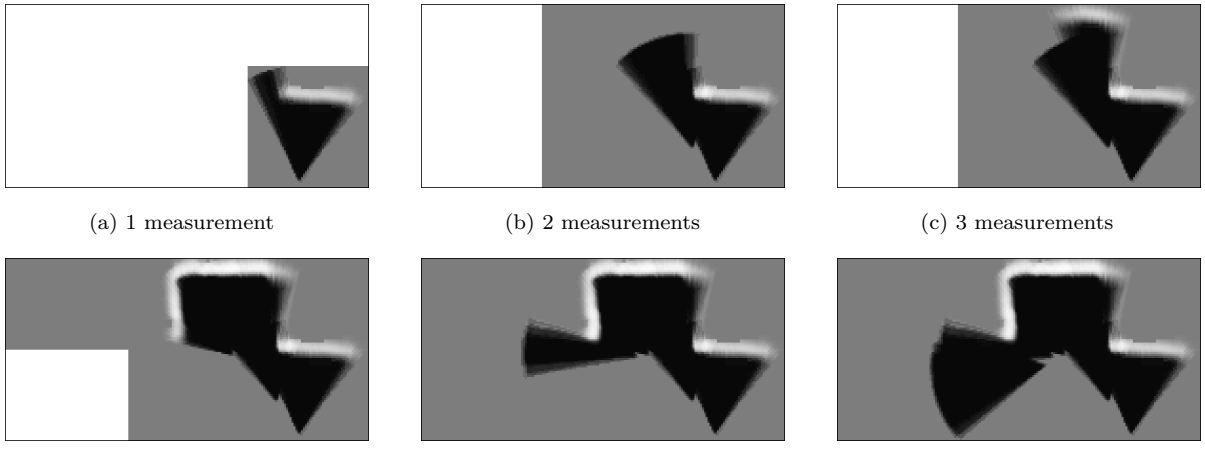
In Figure 6.11 we use the same map as in all examples thus far, but the map size is also handled adaptively as the robot moves through the environment. Here the maps are shown after incorporating measurements 1 to 3 and 6 to 8 in the presence of measurement noise and pose uncertainty.



(a) regular grid (b) adaptive grid (c) adaptive grid

(d) regular grid (e) adaptive grid (f) adaptive grid

Figure 6.10 – Comparing the adaptive occupancy grid map to its regular counterpart after eight measurements in the case of no pose uncertainty ((a)–(c)) and where pose uncertainty is incorporated ((d)–(f)).



(a) 1 measurement (b) 2 measurements (c) 3 measurements

(d) 6 measurements (e) 7 measurements (f) 8 measurements

Figure 6.11 – An example where the total map size is handled adaptively.

Chapter 7

Results

Through the course of the preceding chapters we developed an adaptive occupancy grid mapping algorithm that incorporates both measurement and pose uncertainty. This algorithm can build either 2D or 3D maps, depending on the type of measurements and robot motion.

In this chapter we investigate the performance of our algorithm, first using simulated data and then also using real-world data. The main contribution of this study is in the form of improvements to existing algorithms, and it is therefore important to compare our algorithms to the existing ones. In simulations, where ground truth maps are available, we can evaluate the accuracy of results by plotting ROC curves.

7.1 ROC curves

In order to evaluate the performance of our proposed algorithm, in a simulation environment, a comparison between the generated and ground truth map can be made. A standard way of comparing the performance of different algorithms is by plotting a receiver operating characteristic (ROC) curve [19]. In this section we discuss how to generate and interpret such a curve.

When evaluating the performance of a mapping algorithm, we are interested in its ability to identify occupied cells accurately and to limit the number of false positives. The ROC curve is particularly well suited to depict this trade-off.

The map generated by an occupancy grid algorithm consists of probability values, each indicating the probability that a particular cell is occupied. Every cell in the generated map can therefore be labelled as either occupied or free by applying some fixed threshold to its probability value. This labelling of cells is then compared to the ground truth states.

Consider the table in Figure 7.1, which is known as a contingency table or confusion matrix. Here A indicates the number of cells correctly labelled as occupied (O), while B indicates the number of occupied cells wrongly labelled as free (F). Similarly, C is the number of free cells labelled occupied and D the number of free cells that are correctly labelled.

		Labelling O	Labelling F
Reality	O	A	B
	F	C	D

Figure 7.1 – The confusion matrix of a two class labelling system. The columns indicate the labelling of the variables as O or F and the rows indicate the true states of the variables. The variables A, B, C and D indicate the number of labels in each of the four scenarios for a specific map and threshold.

The true positive ratio (TPR), also known as the sensitivity, is calculated from these variables as

$$\text{TPR} = \frac{A}{A+B}, \tag{7.1.1}$$

and is an indication of how sensitive the system is to detecting occupied cells. A high sensitivity means that the sysem will rarely miss an event when it occurs, which is why high sensitivity is desirable.

The true negative ratio (TNR), or specificity, is calculated as

$$\text{TNR} = \frac{D}{C+D}, \tag{7.1.2}$$

and is an indication of how specific the system is in the labelling of the cells. A high specificity implies that the system has a low rate of false alarms, which is also a desirable characteristic. From this the false positive ratio (FPR) is defined as

$$\begin{aligned} \text{FPR} &= 1 - \text{TNR} \\ &= \frac{C}{C+D}. \end{aligned} \tag{7.1.3}$$

The ROC curve shows the FPR versus the sensitivity, parametrized by all possible thresholds. This is a graphical representation of the compromise made between sensitivity and specificity since an increase in sensitivity, for example, results in a decrease in specificity. Note that all ROC curves start at the point $(0, 0)$ and end at $(1, 1)$.

Ideally, the ROC curve should pass through the point $(0, 1)$ as that would imply perfect detection and no false positives. We can analyze the performance of different systems by the proximity of the curve to the point $(0, 1)$ and also by the area under the curve. In the ideal situation the area under the curve would be 1 which means that the larger the area under the curve, the better the algorithm performs.

7.2 Simulation results

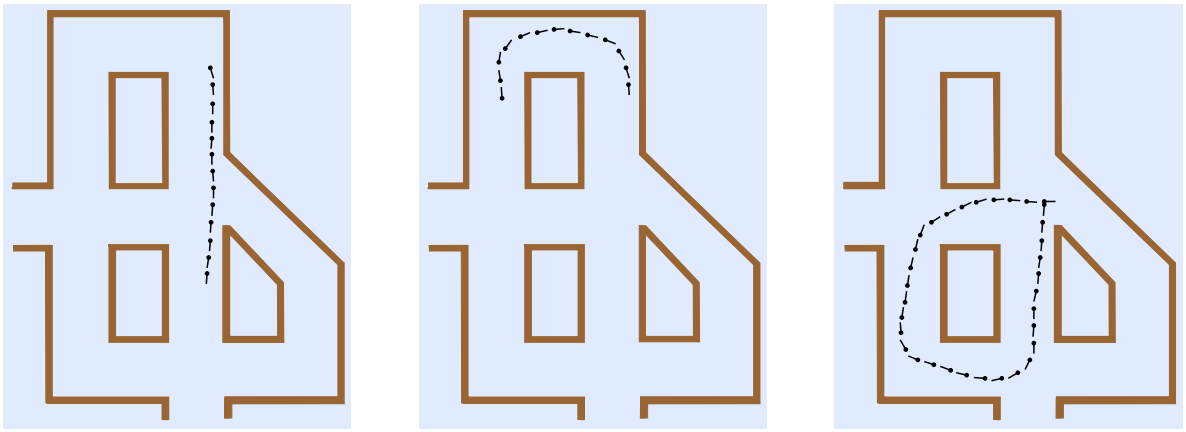
In this section we present results from testing our methods on simulated data. We adapt the simulation environment of Brink [20] that was developed originally as a testing platform for SLAM. An environment is created that typically consists of straight-edge obstacles, and landmarks are defined. A simulated robot then moves through the environment by following set waypoints, and gathers measurements of the landmarks. Noise is added to these measurements, and also to the control commands, to generate inputs for a SLAM system that must estimate the robot's pose as well as the locations of observed landmarks. We equip the robot with a simulated laser scanner that captures range measurements of the environment as the robot moves through it. At every time step we obtain a range measurement and a pose estimate from the SLAM system, together with associated uncertainties, as input for our mapping algorithm.

7.2.1 2D environment

For the first simulation we create a 2D environment and select three different routes through it on which to test our algorithm. Figure 7.2 shows the simulated environment and the three routes. Obstacles are indicated in brown and the robot location at each time instance is indicated by a black dot, with a black line segment indicating its bearing at that time. The robot moves in 2D and is equipped with a 180° field of view laser range scanner. We run both EKF SLAM and FastSLAM on all three datasets.

The results from these datasets follow shortly but let us first discuss general conclusions that are valid for all three datasets as well as the two SLAM algorithms.

Firstly, the performance of the mapping algorithm relies heavily on the accuracy of the pose estimates from the SLAM algorithm, as expected. In Figure 7.3 the performance of the basic occupancy grid algorithm (described in Chapter 3) is given in the presence of increasing pose uncertainty. The uncertainty is increased by adding more noise (via a larger standard deviation σ of the distribution used to simulate noise) to the landmark measurements for the EKF SLAM system. We see that the performance of the algorithm deteriorates with increasing pose uncertainty. As a result the basic occupancy grid algorithm can only perform well if the input pose estimates are reasonably accurate.



(a) corridor dataset (b) turn dataset (c) loop dataset

Figure 7.2 – A simulated 2D environment is divided into three datasets, each representing a different type of robot movement. In each case the true locations of the robot are shown as black dots with line segments corresponding to bearing.

Next we investigate the effect of incorporating pose uncertainty into the map by using our method described in Chapter 5. Consider the corridor dataset and the two different sets of poses shown in Figure 7.4. These two routes have different associated uncertainties: the one in (b) has less uncertainty than the one in (c) and is generally closer to the ground truth route.

The maps obtained from the basic occupancy grid algorithm for these two sets of pose estimates are shown in Figure 7.5(a)–(b) and the maps obtained from our algorithm that incorporate pose uncertainty are shown in (c)–(d).

At first glance it is impossible to see which map in Figure 7.5(a) or (b) is built with more certainty, unless the ground truth is consulted. When (c)–(d) is considered, however, it is clear that (d) carries more uncertainty than (c). If regions with large uncertainty can be detected by a path planner, they can be avoided. This strategy may not be possible when the basic occupancy grid algorithm is used. Also note the small black regions located near the top right of the maps in (a)–(b) that are caused by erroneous range measurements. By incorporating uncertainty, these regions become close to unknown in (c)–(d).

Figure 7.6 shows the ROC curves of the basic occupancy grid algorithm and our algorithm that incorporates pose uncertainty. From these curves we see that their performances are almost similar. Note

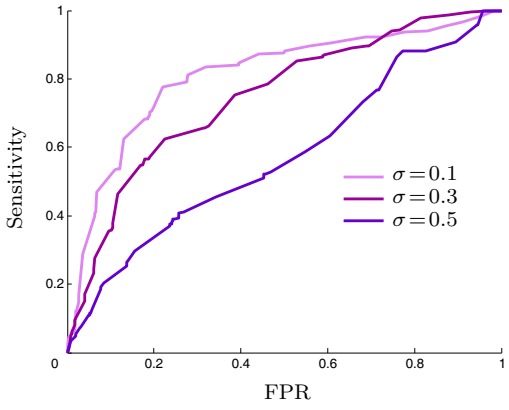
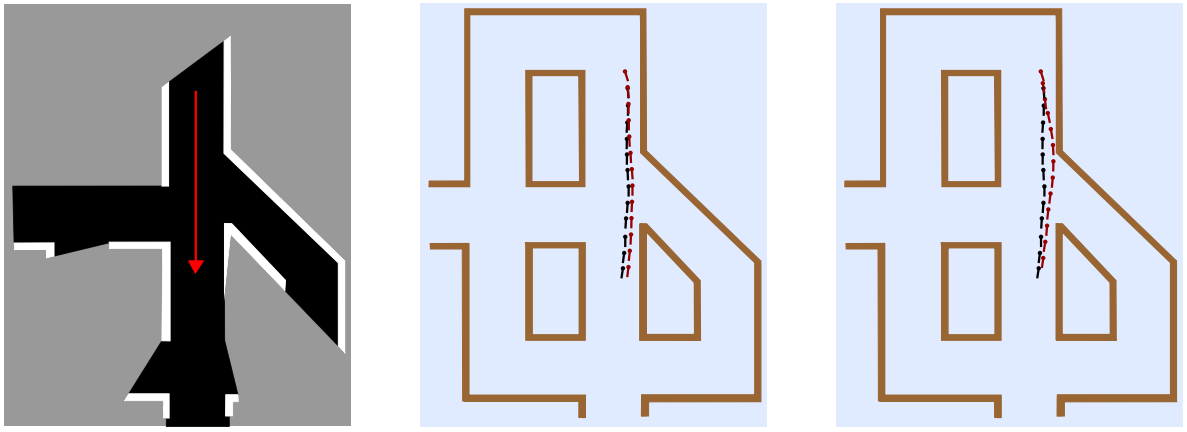


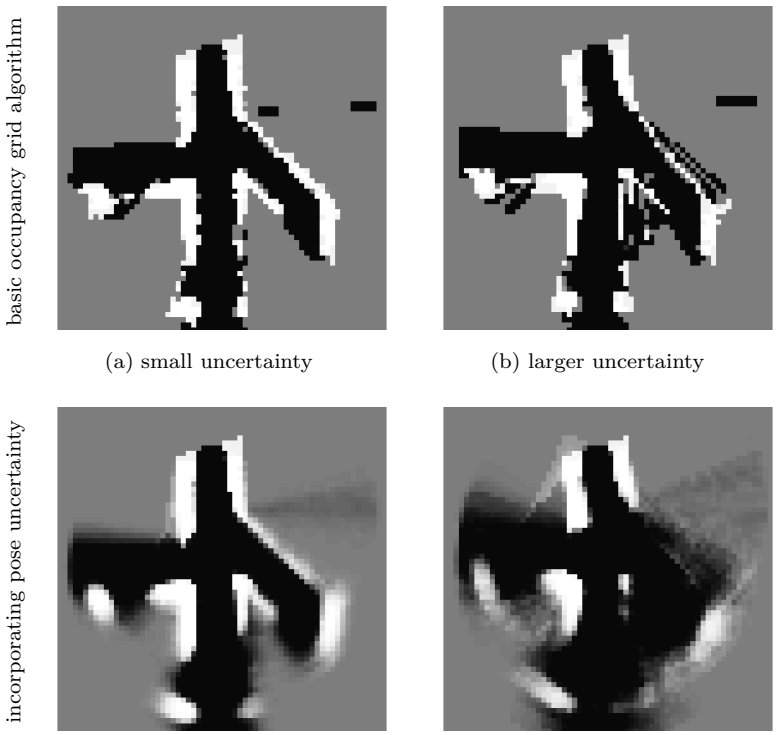
Figure 7.3 – ROC curve depicting the performance of the basic occupancy grid algorithm in the presence of increasing pose uncertainty (resulting from increasing the noise in landmark measurements used for SLAM).



(a) ground truth map (b) path with small uncertainty (c) path with larger uncertainty

Figure 7.4 – The ground truth map and two paths, shown in red in (b) and (c), generated by EKF SLAM for the corridor dataset. The estimated route in (b) has less associated uncertainty than the one in (c).

that we do not expect our algorithm to outperform the basic algorithm, where the most likely pose estimate is used at every time step. By incorporating pose uncertainty we cannot expect to gain accuracy, especially if the most likely pose estimate is inaccurate, but it provides a way to transfer the uncertainty in the robot conditions to the map so that regions with high uncertainty can be distinguished from those with low uncertainty. The fact that our algorithm does not perform significantly worse than the



(a) small uncertainty (b) larger uncertainty

(c) small uncertainty (d) larger uncertainty

Figure 7.5 – Maps obtained from the basic occupancy algorithm, using the pose estimates in Figure 7.4, are shown in (a)–(b), while the maps where pose uncertainty is included are shown in (c)–(d).

basic algorithm implies that by adding pose uncertainty to the map little (if any) accuracy is lost while information regarding uncertainty is gained.

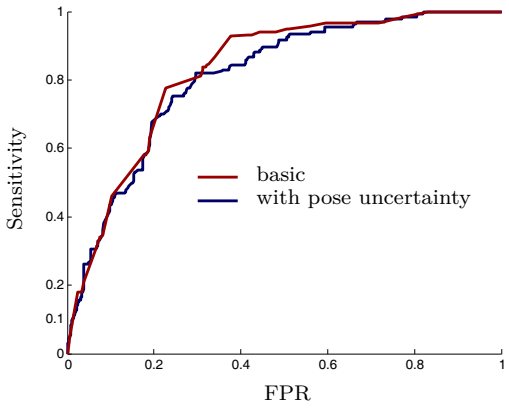


Figure 7.6 – ROC curve depicting the performance of the basic occupancy grid algorithm and our algorithm that incorporates pose uncertainty.

The other key contribution of this work is the improvement made on the adaptive grid mapping algorithm from [37], as discussed in Chapter 6. Specifically, we count not only hits and misses but also unknowns when deciding whether or not to split a grid cell. A typical example of the behaviour of these two algorithms can be seen in Figure 7.7.

(a) regular grid (b) adaptive without unknowns (c) adaptive with unknowns

Figure 7.7 – Comparing the maps from the original adaptive grid mapping algorithm and our improved algorithm with the regular grid mapping algorithm. The regular grid is shown in (a), the adaptive grid map built without counting unknowns in (b) and our adaptive grid map built with unknowns is shown in (c).

In Figure 7.7(a) the regular grid map where pose uncertainty has been incorporated is shown (which is the same as the one in Figure 7.5(c)), in (b) the adaptive map (without gridlines) from the algorithm presented in [37] is shown, while the map from our improved algorithm is shown in (c). Note that the large black block occurring in (b) is absent from (c) so that, in this case, the map from our algorithm resembles the regular grid more closely. The regular grid has 4 096 cells, while the maps in (b) and (c) consist of only 1 717 and 1 861 cells respectively. Although our code is not optimized we found that, on average, the runtime to build a regular grid map is roughly twice that of the adaptive mapping algorithm. With some code optimization we expect this to improve significantly.

We consider an adaptive grid mapping algorithm to be successful if it is able to mimic its regular grid counterpart using fewer cells. Therefore, using the regular grid as the ground truth, the goal is to produce a perfect ROC curve. In Figure 7.8 we compare the adaptive grid algorithm with and without counting unknowns in this manner. Here the areas under the curve for the algorithm without unknowns and the one with unknowns are 0.8917 and 0.9837 respectively. We deduce that our algorithm that keeps track of unknowns outperforms the one without unknowns and is a significant improvement on the original algorithm from [37].

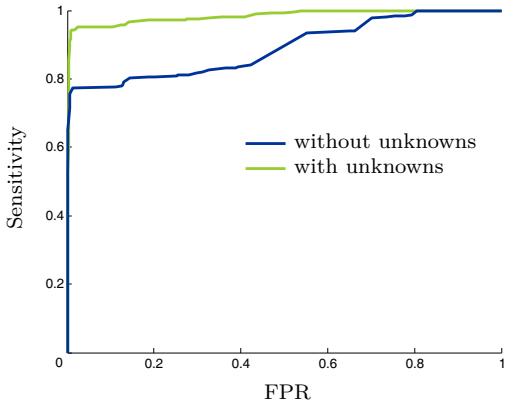
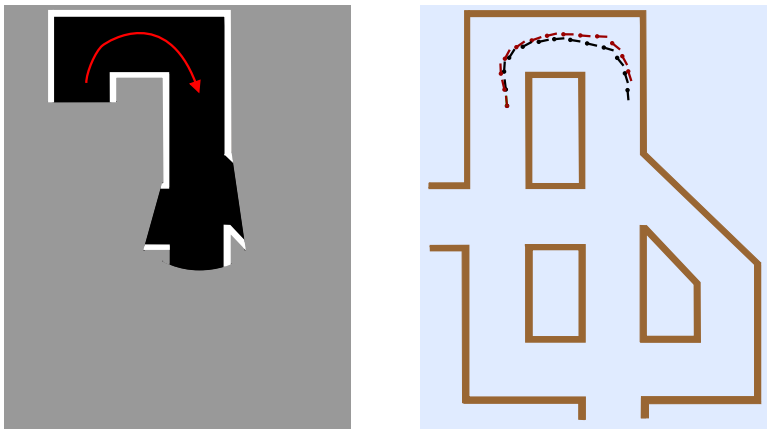


Figure 7.8 – ROC curve comparing the performance of the adaptive grid algorithm, with and without counting unknowns, to the regular grid algorithm.

Next we show maps obtained from the turn and loop datasets. The ground truth map and the route as estimated by EKF SLAM for the turn dataset are shown in Figure 7.9. The odometry of the robot is particularly prone to drift when the vehicle makes a turn and this is reflected in the maps.

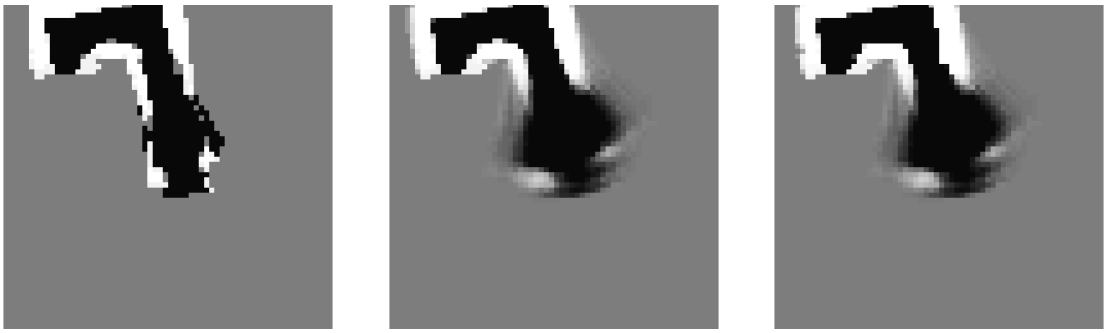


(a) ground truth map (b) estimated path

Figure 7.9 – The route as estimated by EKF SLAM for the turn dataset. In (b) the true path is shown in black, and the SLAM estimation in red.

The map from the basic occupancy grid algorithm applied to the turn dataset is shown in Figure 7.10(a). The map that includes pose uncertainty is shown in (b) and the map from the adaptive grid mapping algorithm which counts unknowns is shown in (c). Here we see that the grid that includes pose uncertainty

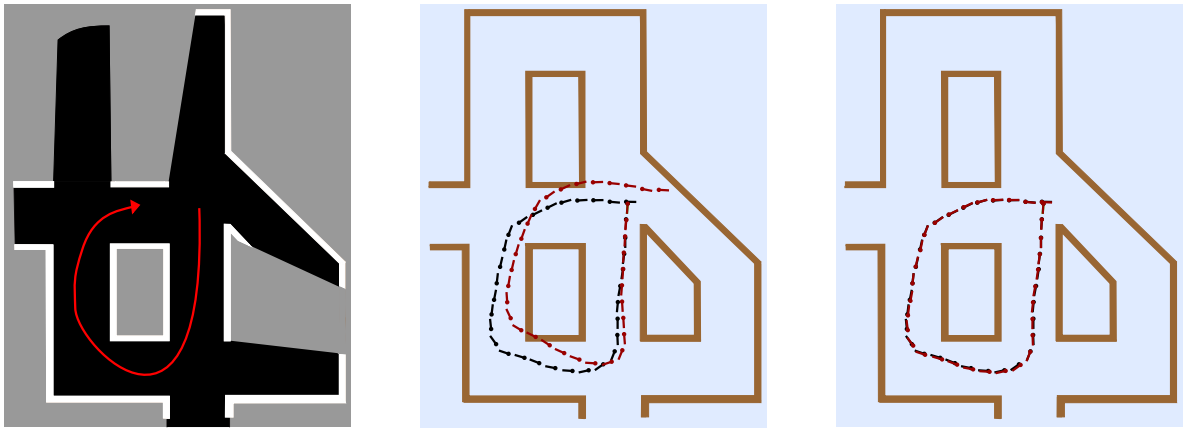
indicates that we have little confidence in certain regions of the map, while the same cannot be said for the map from the basic occupancy grid algorithm. The regular grid in this case has 4 096 cells, while the map with adaptive grid sizes has only 859 cells, and there are no obvious differences between the two.



(a) basic algorithm (b) with pose uncertainty (c) adaptive grid

Figure 7.10 – Occupancy grid maps obtained for the turn dataset. In (a) the map from the basic algorithm is shown, while (b) shows the regular grid map with pose uncertainty. The map from our adaptive grid mapping algorithm is shown in (c).

For the loop dataset we show two sets of maps in Figure 7.11, one for an EKF SLAM path estimate and one for a FastSLAM path estimate. In this case the FastSLAM path is much closer to the true path than the EKF path, and carries less uncertainty.



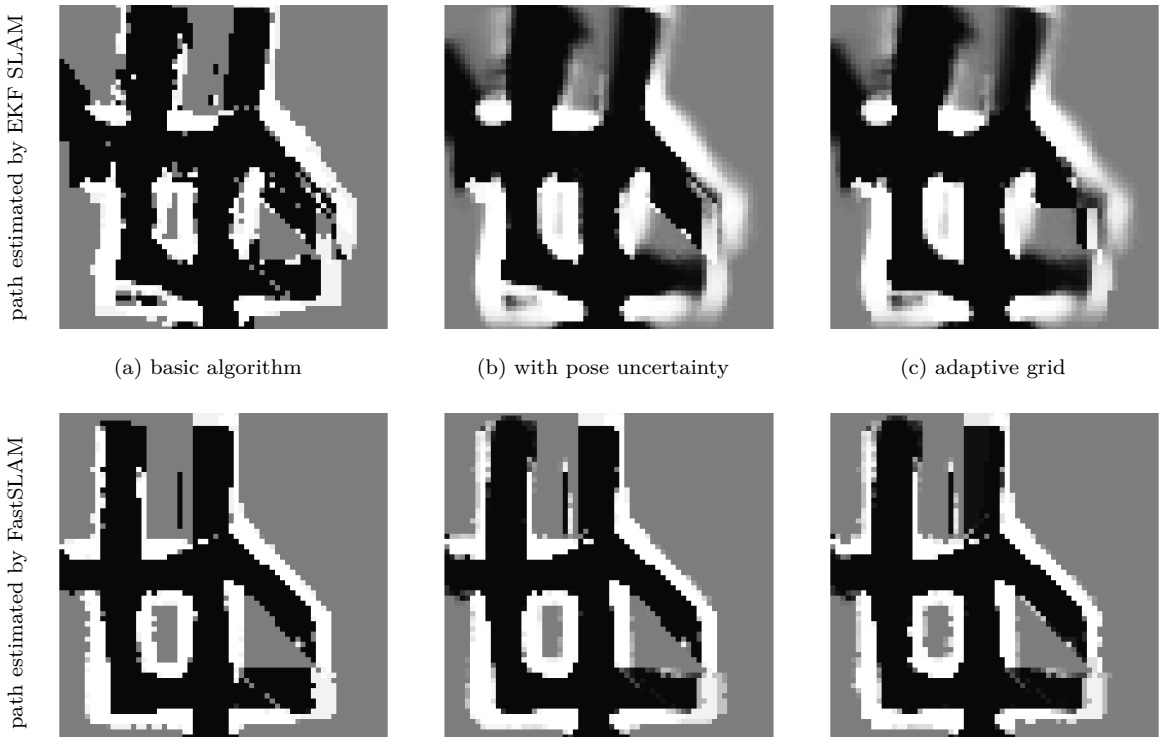
(a) ground truth map (b) EKF path estimate (c) FastSLAM path estimate

Figure 7.11 – For the loop dataset we have the ground truth map in (a), and the estimated route as obtained from an EKF SLAM algorithm and a FastSLAM algorithm. Again, the true path is shown in black and the estimated paths in red.

The maps obtained from the three different mapping algorithms for the EKF path are shown in Figure 7.12(a)–(c). Here the adaptive grid has 2 104 cells which is still significantly less than the 4 096 of the regular grid. Between these two maps there is a notable difference in the lower right region where a part of the adaptive grid map has probability values close to unknown, while the regular grid shows it as free. This phenomenon can be explained by the slight bias introduced by the adaptive algorithm's

tendency to consider a cell as unknown in the absence of sufficient information. This results in a more conservative algorithm which is preferable to an overconfident one.

In the case of FastSLAM the effect of including pose uncertainty into the map is much less apparent than for the EKF SLAM case. The map from the basic algorithm, the map where pose uncertainty is included in the regular grid as well as our adaptive grid are shown in Figure 7.12(d)–(f).



(a) basic algorithm (b) with pose uncertainty (c) adaptive grid

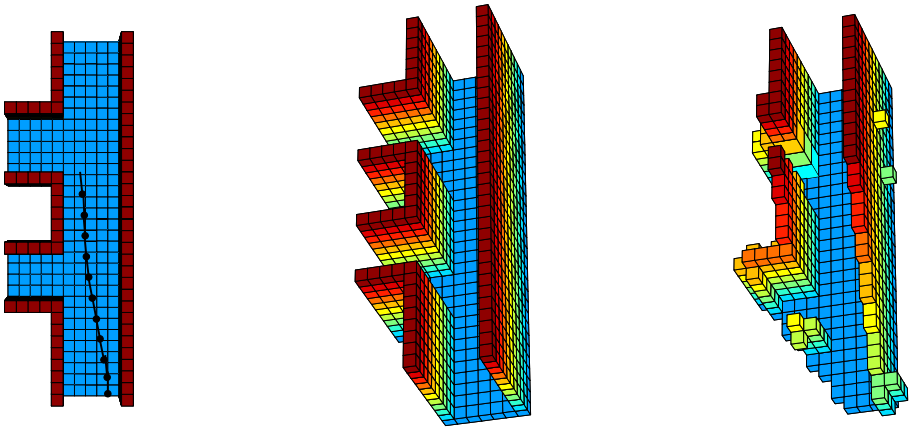
(d) basic algorithm (e) with pose uncertainty (f) adaptive grid

Figure 7.12 – Occupancy grid maps obtained for the loop dataset and the EKF SLAM path are shown in (a)–(c) while the maps for the FastSLAM path are shown in (d)–(f). In (a) and (d) the maps from the basic algorithm are shown, while (b) and (e) show the regular grid maps with pose uncertainty. The maps from our adaptive grid mapping algorithm are shown in (c) and (f).

7.2.2 3D environment

We also present results from a simple 3D simulation where we have a sensor that generates a 480×640 range image at each time step. The simulated environment and the estimated route through it are shown in Figure 7.13(a). Here only obstacles are shown and are coloured according to their height from the ground plane. In (c) the map from our adaptive occupancy grid algorithm with pose uncertainty is shown, where only cells labelled as occupied (according to a probability threshold of 0.5) are drawn for clarity. The ground truth map is shown in (b). For this example the FastSLAM algorithm carries so little uncertainty that there is almost no difference between the basic occupancy grid algorithm and the one that includes pose uncertainty, and the former is therefore not included here.

Note that, except for the wall on the lower left, there is little difference between the ground truth map and the one built by our algorithm. The difference in the region mentioned is not observed by the sensor while following the path and that wall is therefore only partially reconstructed.



(a) environment and route (b) ground truth map (c) occupancy grid map

Figure 7.13 – A simulated 3D environment and the corresponding 3D occupancy grid map. The environment and the route followed through it are shown from above in (a), while the ground truth map and the adaptive occupancy grid map with pose uncertainty are shown from the same viewpoint in (b) and (c) respectively.

7.3 Results from real-world data

Finally we present some occupancy grid maps built from data captured by a real robot.

The RAWSEEDS initiative (www.rawseeds.org) provides a number of datasets which comprise of odometry information and outputs of various sensors mounted on a wheeled robot. We select one of the indoor datasets, namely the Bicocca 2009–02–25b set, in which the robot shown in Figure 7.14 drives through a library. Among its many sensors the robot has a pair of stereo cameras and a laser scanner facing forward. Sample images captured by one of these cameras are given in Figure 7.15.

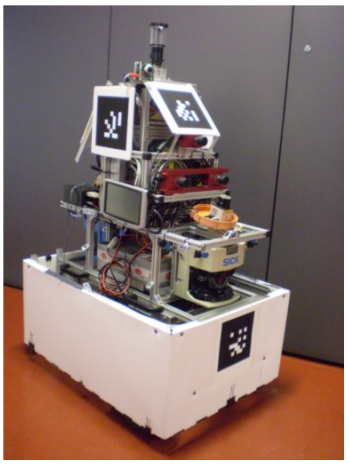


Figure 7.14 – The robot used to capture the RAWSEEDS indoor Bicocca dataset (image from www.rawseeds.org). Note the red pair of stereo cameras and the forward facing laser scanner.

We employ systems by Brink et al. [21; 22] to perform SLAM with the odometry and landmark features found and matched across the stereo image pairs. Output from the forward facing laser scanner, which has a 180° field of view, is used with the pose estimates from the SLAM system to perform occupancy grid mapping. For these tests we implement our complete 2D mapping algorithm (with measurement

(a) (b) (c)

(d) (e) (f)

Figure 7.15 – Sample images captured by the robot for the Bicocca dataset. The positions where these images were captured are indicated on the schematic representations that follow shortly.

and pose uncertainty, our adaptive grid size technique that counts unknowns, and the adaptive total map size extension).

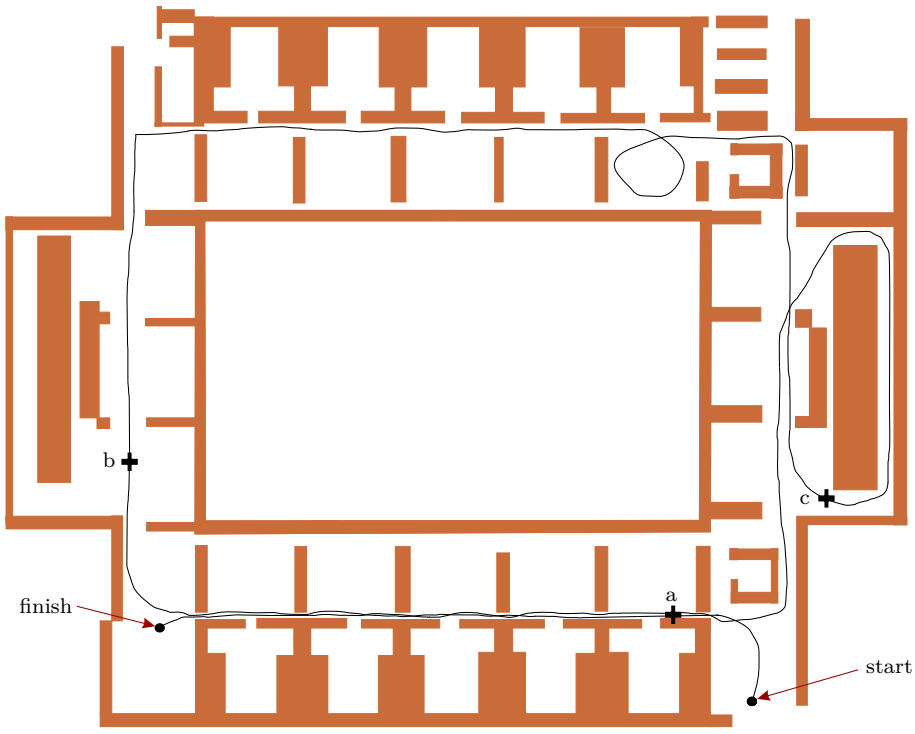
For this dataset we have access to neither the exact trajectory of the robot nor a ground truth map against which we can evaluate our occupancy grid maps. However, a crude schematic representation of the building's interior is available with the dataset which allows for qualitative evaluation.

For illustration purposes we proceed to show and discuss results from two separate sections of the dataset.

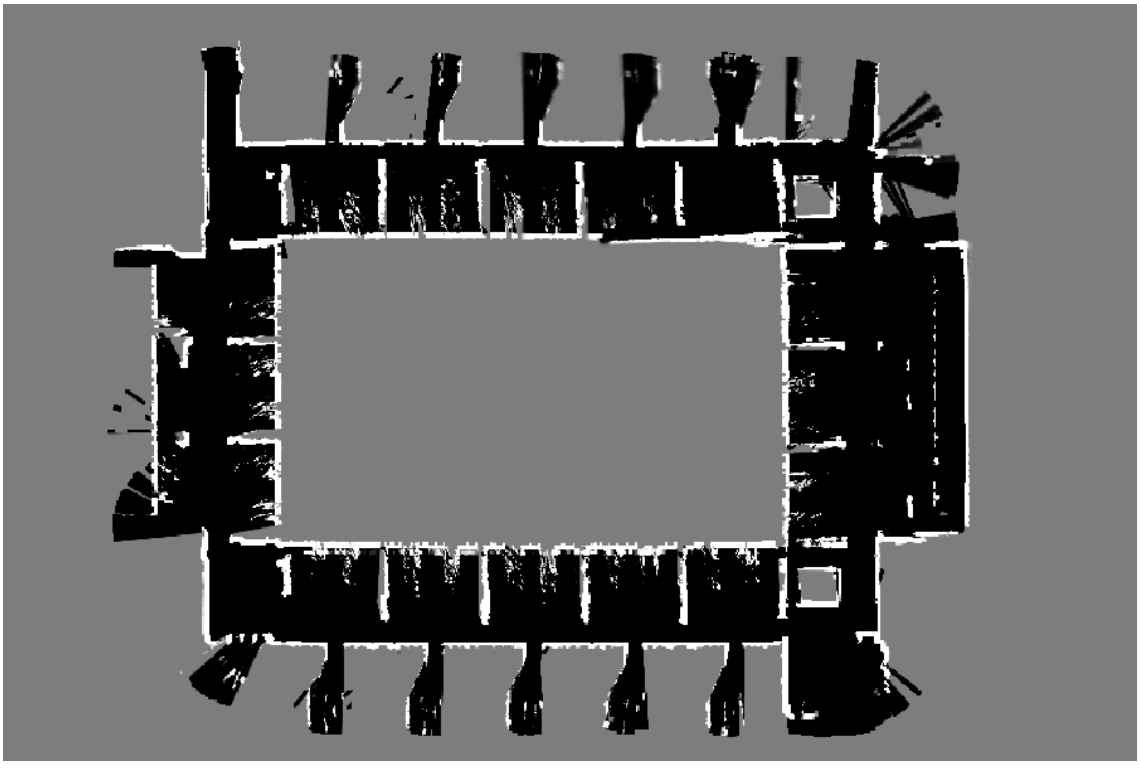
Figure 7.16 shows a schematic representation of the first section, as well as a rough indication of the route followed by the robot, in (a). The robot makes a large loop, and two smaller loops along the way, through part of the library. Note that it also drives through the corridor shown on the bottom for a second time. The pose estimates which we obtain from the FastSLAM 1.0 system of Brink et al. remains remarkably stable throughout this journey of about 275 metres. The occupancy grid map built by our algorithm is shown in (b), and bears near resemblance to the schematic of the building. The environment contains some glass doors (not indicated in the schematic) close to the robot's starting point which were missed by the occupancy grid algorithm. However, the laser range scanner is unable to detect glass obstacles and we therefore cannot expect them to appear in the map.

Results of a second section of the dataset are shown in Figure 7.17. Here the robot drives through a narrow corridor, makes a 90° turn in a small room, drives through a second narrow corridor, maneuvers through a large open space and then drives through another long corridor. The total distance travelled is about 180 metres. The map shown in (b) is obtained with pose estimates from FastSLAM 1.0 which, unfortunately, suffers form a small orientation error made during the first turn and is not able to estimate the pose uncertainty accordingly. The result shown in (c) is obtained with pose estimates from FastSLAM 2.0 and is much closer to the schematic.

The results presented in this chapter stress the fact that, as long as the SLAM system can provide consistent information on the uncertainty in its estimates, our extension to the basic occupancy grid algorithm will be able to incorporate this uncertainty into the map. Furthermore, we demonstrated that our modification to the algorithm proposed in [37] is indeed an improvement and by employing such an adaptive grid mapping algorithm we are able to save both memory and computation time.

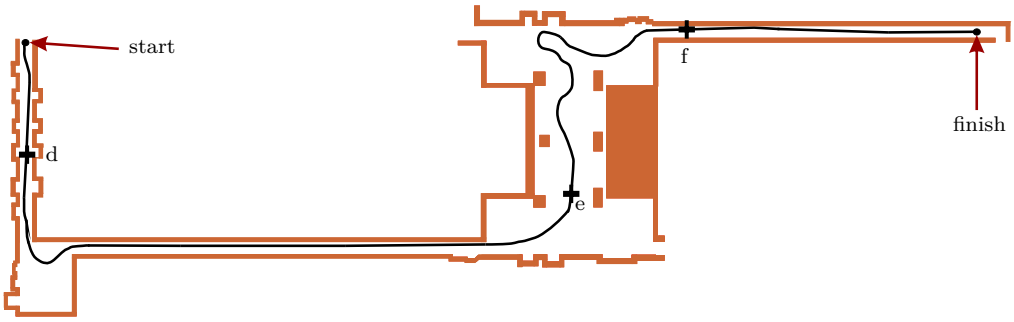


(a) schematic representation of the environment and path followed by the robot

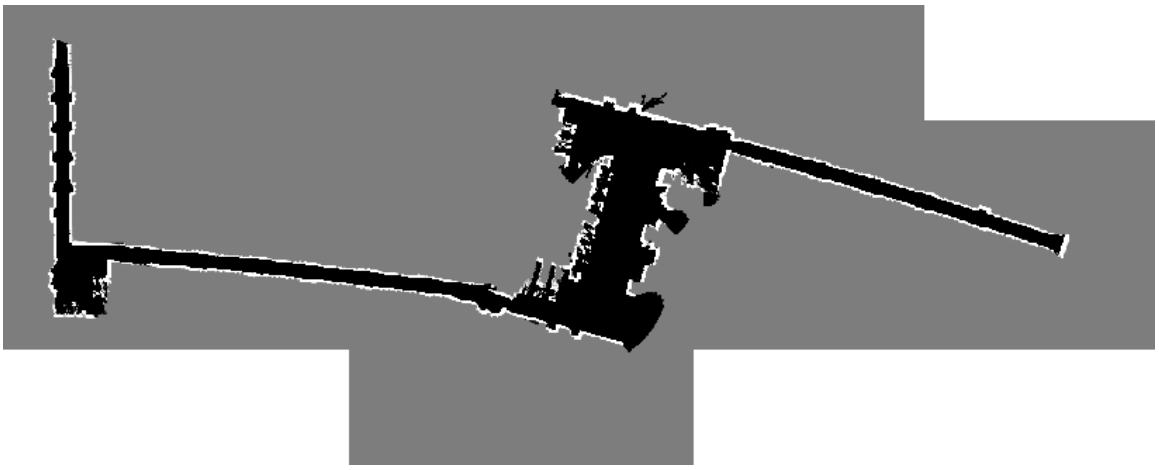


(b) occupancy grid map using pose estimates from FastSLAM 1.0

Figure 7.16 – A schematic representation of the environment and an occupancy grid map built from a subset of the RAWSEEDS Bicocca indoor dataset. In (a) the positions where the images shown in Figure 7.15(a)–(c) are captured are indicated by the crosses and are labelled accordingly.



(a) schematic representation of the environment and path followed by the robot



(b) occupancy grid map using pose estimates from FastSLAM 1.0

(c) occupancy grid map using pose estimates from FastSLAM 2.0

Figure 7.17 – A schematic representation of the environment and the occupancy grid maps built from another subset of the RAWSEEDS Bicocca indoor dataset. In (a) the positions where the images shown in Figure 7.15(d)–(f) are captured are indicated by the crosses and are labelled accordingly. FastSLAM 1.0 introduces an orientation error during the first turn and, since the mapping algorithm relies on pose estimates from the SLAM system, this error is manifested in the map.

Chapter 8

Conclusions and future work

The main focus of this thesis was on robotic mapping to promote planning and navigation of a fully autonomous vehicle. More specifically, we focussed on the problem of effectively including measurement and pose uncertainty into the widely used occupancy grid mapping algorithm. This incorporation is believed to be essential for successful planning and navigation.

In this chapter we make some concluding remarks and offer ideas for future work.

8.1 Conclusions

Through a review of the relevant literature (in Chapter 1) we saw that the occupancy grid algorithm is state-of-the-art when it comes to robotic mapping algorithms. We then set out to address two points of criticism: the inability of the algorithm to model pose uncertainty during the mapping process as well as the demanding memory requirements.

We started the work by investigating SLAM algorithms for providing the mapping algorithm with pose estimates. These estimates are typically also accompanied by an uncertainty distribution which can be represented by parameters of a closed form pdf or as a set of samples. If we wish to include pose uncertainty into the map, our algorithm has to accommodate both of these uncertainty representations.

Next we introduced the basic occupancy grid algorithm. Here we noted that an important independence assumption is made to make the algorithm computationally tractable but it can lead to inconsistencies in the map. However, we saw that if we choose our grid size appropriately, we can avoid such inconsistencies.

Incorporating measurement uncertainty into the map was discussed next. We saw that an ideal sensor model is inappropriate when the range measurement of the environment is noisy, which is usually the case. The Gaussian inverse sensor model is more suitable in this situation. Although many authors merely state or depict their inverse sensor model we derived ours analytically. Our Gaussian inverse sensor model (derived by convolving two functions f and g) is a function of the distance r to the sensor, the distance Z to the detected obstacle and the standard deviation σ associated with this measurement. It is defined piecewise and is given by

$$(f*g)(r) = \begin{cases} 0, & r \in \left(-\infty, Z-\frac{L}{2}\right), \\ -\frac{1}{2}\mathrm{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) + \frac{1}{2}\mathrm{erf}\left(\frac{r-2Z+\frac{L}{2}}{\sqrt{2}\sigma}\right), & r \in \left[Z-\frac{L}{2}, Z+\frac{L}{2}\right), \\ -\frac{1}{4}\mathrm{erf}\left(\frac{r-2Z-\frac{L}{2}}{\sqrt{2}\sigma}\right) + \frac{1}{2}\mathrm{erf}\left(\frac{r-2Z+\frac{L}{2}}{\sqrt{2}\sigma}\right) - \frac{1}{4}\mathrm{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right), & r \in \left[Z+\frac{L}{2}, \infty\right). \end{cases} \tag{8.1.1}$$

We also investigated tailoring the Gaussian inverse sensor model for a specific sensor and provided details for three types of sensors: stereo cameras, structured light sensors and time-of-flight sensors.

A further contribution of this work enables the modelling of pose uncertainty in the mapping process. We altered the update equation of the basic occupancy grid algorithm, which is given by

$$\log\left(\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})}\right) = \log\left(\frac{p(m_i|z_t)}{p(m_i^c|z_t)}\right) + \log\left(\frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})}\right) - \log\left(\frac{p(m_i)}{p(m_i^c)}\right), \tag{8.1.2}$$

to incorporate weighted samples of the pose uncertainty distribution according to

$$\log\left(\frac{p(m_i|z_{1:t})}{p(m_i^c|z_{1:t})}\right) = \sum_{j=1}^{M} w_t^{[j]} \log\left(\frac{p\left(m_i \left| z_t^{[j]}\right.\right)}{p\left(m_i^c \left| z_t^{[j]}\right.\right)}\right) + \log\left(\frac{p(m_i|z_{1:t-1})}{p(m_i^c|z_{1:t-1})}\right) - \log\left(\frac{p(m_i)}{p(m_i^c)}\right), \tag{8.1.3}$$

where m_i is the event that cell i is occupied, m_i^c is the event that the cell is free and z_t is the measurement at time t. If the uncertainty distribution is provided in closed form, we first sample from it. This approach seems to incorporate pose uncertainty into the map effectively and, if an erroneous or noisy measurement is received, the effect on the map is much less severe than for the basic occupancy grid algorithm. It also enables a navigation module to distinguish between regions in the map that were built with high certainty from those built with low certainty. Our simulation results corroborate the conclusion that we managed to find a way of including pose uncertainty into the map without sacrificing accuracy.

The other main contribution of this work was to decrease memory consumption and increase efficiency. This was achieved by implementing an algorithm that can adapt its grid size locally on-the-fly by representing the environment to be mapped by a region tree. This idea was first proposed in [37] where the decision to split a cell into smaller cells is based on the number of hits and misses it receives from the measurements. However, we saw that a cell can be partially observed by some measurement beams while being obscured from view to other beams. We proposed to base the decision of splitting a cell on the number of hits, misses and unknowns that it receives. This involves computing

$$X^2_{m_i} = \frac{(n_o - E_{n_o,m_i})^2}{E_{n_o,m_i}} + \frac{(n_f - E_{n_f,m_i})^2}{E_{n_f,m_i}} + \frac{(n_u - E_{n_u,m_i})^2}{E_{n_u,m_i}}, \tag{8.1.4}$$

$$X^2_{m_i^c} = \frac{(n_o - E_{n_o,m_i^c})^2}{E_{n_o,m_i^c}} + \frac{(n_f - E_{n_f,m_i^c})^2}{E_{n_f,m_i^c}} + \frac{(n_u - E_{n_u,m_i^c})^2}{E_{n_u,m_i^c}}, \tag{8.1.5}$$

$$X^2_{m_i^u} = \frac{(n_o - E_{n_o,m_i^u})^2}{E_{n_o,m_i^u}} + \frac{(n_f - E_{n_f,m_i^u})^2}{E_{n_f,m_i^u}} + \frac{(n_u - E_{n_u,m_i^u})^2}{E_{n_u,m_i^u}}, \tag{8.1.6}$$

where n_o, n_f and n_u denote the number of hits, misses and unknowns respectively, m_i^u is the event that the cell is unknown (obscured from view), and E indicates the expected value where the first subscript refers to the number of hits, misses or unknowns and the second to the true state of the cell. If

$$\min\left(X^2_{m_i}, X^2_{m_i^c}\right) > \chi^2_{2,1-\alpha}, \tag{8.1.7}$$

where $\chi^2_{2,1-\alpha}$ is set to 10.597, and

$$X^2_{m_i^u} > \chi^2_{2,1-\alpha}, \tag{8.1.8}$$

the cell is split. However, if (8.1.8) is not satisfied, no new probability value is assigned to the cell since it has not received sufficient measurements. Adaptive grid mapping generally results in far fewer cells than regular grid mapping, and our new split test significantly improves the ability of the adaptive algorithm to mimic its regular grid counterpart.

Furthermore, we showed that the algorithm is feasible in both 2D and 3D, and the extension from 2D to 3D in all aspects of the algorithm was found to be fairly straightforward. We also demonstrated that our algorithm functions well in both simulated and real environments, and behaves as the theory would suggest.

8.2 Future work

The problem of robotic mapping remains an active research area and, with our contributions to the occupancy grid algorithm taken into consideration, offers many directions for further development. We proceed to list some ideas on how the algorithm can be improved and generalized.

At the start of Chapter 5 we mentioned that it might be possible (at least in theory) to incorporate pose uncertainty by blurring the occupancy grid map with an appropriate Gaussian kernel. This is applicable if the pose distribution is Gaussian, as in EKF SLAM, but can present difficulties in handling

the dimensions of the kernel that correspond to the robot's bearing. We believe that this approach warrants further investigation. In fact, it can be considered a special case of the more general problem of calculating the exact value of (5.4.6), i.e. the expected log odds ratio of a particular cell, given the range measurements and any closed-form pdf describing the pose uncertainty.

We stated in Chapter 7 that our implementation is not optimized for efficiency in terms of execution time. We set out to validate the theoretical arguments behind our approach, so the conclusions drawn from our results remain valid, but there is room for improving the implementation. We note that our new update equation, stated in (8.1.3), can lead to a slightly different algorithmic structure where, for every cell, a single assignment is made from all the measurement beams that intersect the cell across all pose samples. Furthermore, the incorporation of many pose samples into a map consisting of many independent cells lends itself to effective parallelization across multiple cores.

Many range sensors can operate at relatively high frequencies. The laser range scanner used in section 7.3, for example, returns measurements at 75 Hz. Using all this information can introduce redundancy at the expense of unnecessary computation, particularly if the robot moves at a relatively slow speed. A selective approach that decides on-the-fly which measurements to incorporate into the map, by balancing new information gained against the size of overlap with previously seen regions, offers an interesting research problem.

The employment of multiple range sensors, and the fusion of maps generated by each individual sensor, is currently a norm in autonomous navigation. However, in the context of adaptive grid mapping, different maps of the same area cannot be compared on a cell level due to possibly different cell sizes. Sensor fusion may therefore not be straightforward.

A further expansion of the work presented in this thesis can be to use the maps generated by our occupancy grid algorithm as feedback to the SLAM system. The idea of using occupancy grid maps for localization refinement has been investigated [82] and it would be interesting to see how our incorporation of pose uncertainty affects the design and performance of those methods.

One aspect of robotic mapping that requires attention is the problem of handling dynamic environments. The static environment assumption made in this thesis allows for a simple update equation but, when moving objects are present in the environment, the map will become inconsistent. The idea of using our improved occupancy grid algorithm for mapping dynamic environments, coupled with the challenging task of performing SLAM in a dynamic environment, seems like a promising direction for further study.

Since a solution to the mapping problem would ultimately be a component in some larger autonomous robotic system, it is important to further investigate the feasibility of the maps generated by our algorithm for tasks such as path planning, obstacle detection and avoidance, and navigation in general.

We believe that by presenting the work in this thesis some progress has been made towards solving the robotic mapping problem and developing a fully autonomous vehicle.

Bibliography

[1] J. Amanatides, A. Woo, *A fast voxel traversal algorithm for ray tracing*, Eurographics, pp. 3–10, 1987.

[2] F. Andert, *Drawing stereo disparity images into occupancy grids: measurement model and fast implementation*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5191–5197, 2009.

[3] J. Aulinas, Y. Petillot, J. Salvi, X. Lladó, *The SLAM problem: a survey*, International Conference of the Catalan Association for Artificial Intelligence, pp. 363–371, 2008.

[4] T. Bailey, J. Nieto, J. Guivant, M. Stevens, E. Nebot, *Consistency of the EKF-SLAM algorithm*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3562–3568, 2006.

[5] C. Balaguer, A. Gimenez, A. Jardon, R. Cabas, R. Correal, *Live experimentation of the service robot applications for elderly people care in home environments*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2345–2350, 2005.

[6] L. Banjanović-Mehmedović, I. Petrović, E. Ivanjko, *Mobile robot localization using local occupancy grid maps transformations*, Power Electronics and Motion Control Conference, pp. 1307–1312, 2006.

[7] T. Barfoot, *Online visual motion estimation using FastSLAM with SIFT features*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 579–585, 2005.

[8] H. Bay, A. Ess, T. Tuytelaars, L. van Gool, *Speeded-up robost features (SURF)*, Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, 2008.

[9] J. Beraldin, *Integration of laser scanning and close-range photogrammetry - the last decade and beyond*, International Society for Photogrammetry and Remote Sensing Congress, pp. 972–983, 2004.

[10] P. Besl, N. McKay, *A method for registration of 3D shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239–256, 1992.

[11] A. Birk, S. Carpin, *Merging occupancy grid maps from multiple robots*, Proceedings of the IEEE, vol. 94, no. 7, pp. 1384–1397, 2006.

[12] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[13] R. Biswas, B. Limketkai, S. Sanner, S. Thrun, *Towards object mapping in dynamic environments with mobile robots*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1014–1019, 2002.

[14] F. Blais, *Review of 20 years of range sensor development*, Journal of Electronic Imaging, vol. 13, no. 1, pp. 231–240, 2004.

[15] F. Blais, J. Beraldin, S. El-Hakim, *Range error analysis of an integrated time-of-flight, triangulation and photogrammetry 3D laser scanning system*, SPIE International Aerosense Symposium, pp. 236–247, 2000.

[16] F. Bonin-Font, A. Ortiz, G. Oliver, *Visual navigation for mobile robots: a survey*, Journal of Intelligent Robotic Systems, vol. 53, no. 3, pp. 263–296, 2008.

[17] J. Borenstein, Y. Koren, *The vector field histogram – fast obstacle avoidance for mobile robots*, IEEE Journal of Robotics and Automation, vol. 7, no. 3, pp. 278–288, 1991.

[18] M. Bouzouraa, U. Hofmann, *Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors*, IEEE Intelligent Vehicles Symposium, pp. 294–300, 2010.

[19] A. Bradley, *The use of the area under the ROC curve in the evaluation of machine learning algorithms*, Pattern Recognition, vol. 30, no. 7, pp. 1145–1159, 1997.

[20] W. Brink, *Stereo vision for simultaneous localization and mapping*, MScEng thesis, Stellenbosch University, 2012.

[21] W. Brink, C. van Daalen, W. Brink, *Stereo vision as a sensor for EKF SLAM*, Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa, pp. 19–24, 2011.

[22] W. Brink, C. van Daalen, W. Brink, *Probabilistic outlier removal for robust landmark identification in stereo vision based SLAM*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2822–2827, 2012.

[23] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun, *Experiences with an interactive museum tour-guide robot*, Artificial Intelligence, vol. 114, no. 1–2, pp. 3–55, 1999.

[24] S. Carpin, *Fast and accurate map merging for multi-robot systems*, Autonomous Robots, vol. 25, no. 3, pp. 305–316, 2008.

[25] J. Castellanos, J. Montiel, J. Neira, J. Tardós, *The SPmap: a probabilistic framework for simultaneous localization and map building*, IEEE International Conference on Robotics and Automation, pp. 948–953, 1999.

[26] R. Chatila, J. Laumond, *Position referencing and consistent world modeling for mobile robots*, IEEE International Conference on Robotics and Automation, vol. 2, pp. 138–145, 1985.

[27] M. Cikes, M. Dakulovic, I. Petrović, *The path planning algorithms for a mobile robot based on the occupancy grid map of the environment – a comparative study*, International Symposium on Information, Communication and Automation Technologies, pp. 1–8, 2011.

[28] A. Clemente, A. Davison, I. Reid, J. Neira, J. Tardós, *Mapping large loops with a single hand-held camera*, Robotics: Science and Systems, pp. 297–304, 2007.

[29] C. Connolly, *Cumulative generation of octree models from range data*, IEEE International Conference on Robotics and Automation, pp. 25–32, 1984.

[30] C. Crassin, F. Neyret, S. Lefebvre, E. Eisemann, *GigaVoxels: ray-guided streaming for efficient and detailed voxel rendering*, Symposium on Interactive 3D Graphics and Games, pp. 15–22, 2009.

[31] B. Curless, M. Levoy, *A volumetric method for building complex models from range images*, Proceedings of SIGGRAPH, pp. 302–312, 1996.

[32] R. Danescu, F. Oniga, S. Nedevschi, *Particle grid tracking system for stereovision based environment perception*, IEEE Intelligent Vehicles Symposium, pp. 987–992, 2010.

[33] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, *A solution to the simultaneous localization and map building (SLAM) problem*, IEEE Transactions of Robotics and Automation, vol. 17, no. 3, pp. 229–241, 2001.

[34] I. Dryanovski, W. Morris, J. Xiao, *Multi-volume occupancy grids: an efficient probabilistic 3D mapping model for micro aerial vehicles*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1553–1559, 2010.

[35] H. Durrant-Whyte, T. Bailey, *Simultaneous localization and mapping: part I*, IEEE Robotics and Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.

[36] H. Durrant-Whyte, T. Bailey, *Simultaneous localization and mapping: part II*, IEEE Robotics and Automation Magazine, vol. 13, no. 3, pp. 108–117, 2006.

[37] E. Einhorn, C. Schröter, H. Gross, *Finding the adequate resolution for grid mapping – cell size locally adapting on-the-fly*, IEEE International Conference on Robotics and Automation, pp. 1843–1848, 2011.

[38] A. Elfes, *Using occupancy grids for mobile robot perception and navigation*, IEEE Computer, vol. 22, no. 6, pp. 46–57, 1989.

[39] S. Engelson, D. McDermott, *Error correction in mobile robot map learning*, IEEE International Conference on Robotics and Automation, pp. 2555–2560, 1992.

[40] R. Eustice, H. Singh, J. Leonard, M. Walter, R. Ballard, *Visually navigating the RMS Titanic with SLAM information filters*, Robotics: Science and Systems, pp. 57–64, 2005.

[41] G. Ferri, M. Jakuba, A. Mondini, V. Mattoli, B. Mazzolai, D. Yoerger, *Mapping multiple gas/odor sources in an uncontrolled indoor environment using a Bayesian occupancy grid mapping based method*, Journal of Robotics and Autonomous Systems, vol. 59, no. 11, pp. 988–1000, 2011.

[42] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd edition, Academic Press, 1990.

[43] C. Fulgenzi, A. Spalanzani, C. Laugier, *Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid*, IEEE International Conference on Robotics and Automation, pp. 1610–1616, 2007.

[44] R. Garcia, O. Aycard, T. Vu, *High level sensor data fusion for automotive applications using occupancy grids*, International Conference on Control, Automation, Robotics and Vision, pp. 530–535, 2008.

[45] S. Goldberg, M. Maimone, L. Matthies, *Stereo vision and rover navigation software for planetary exploration*, IEEE Aerospace Conference, pp. 2025–2036, 2002.

[46] J. Guivant, E. Nebot, *Optimization of the simultaneous localization and map-building algorithm for real-time implementation*, IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 242–257, 2001.

[47] V. Gupta, G. Singh, A. Gupta, A. Singh, *Occupancy grid mapping using artificial neural networks*, International Conference on Industrial Electronics, Control and Robotics, pp. 247–250, 2010.

[48] J. Gutmann, M. Fukuchi, M. Fujita, *3D perception and environment map generation for humanoid robot navigation*, International Journal of Robotics Research, vol. 27, no. 10, pp. 1117–1134, 2008.

[49] D. Guzzoni, A. Cheyer, L. Julia, K. Konolige, *Many robots make short work*, Artificial Intelligence Magazine, vol. 18, no. 1, pp. 55–64, 1997.

[50] R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2003.

[51] S. Hrabar, *An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance*, Journal of Field Robotics, vol. 29, no. 2, pp. 215–239, 2012.

[52] E. Ivanjko, I. Petrović, *Extended Kalman filter based mobile robot pose tracking using occupancy grid maps*, IEEE Mediterranean Electrotechnical Conference, pp. 311–314, 2004.

[53] A. Jacoff, E. Messina, J. Evans, *A standard test course for urban search and rescue robots*, PerMIS Workshop: Measuring the Performance and Intelligence of Systems, pp. 252–259, 2000.

[54] M. Jones, J. Maerentzen, M. Sramek, *3D distance fields: a survey of techniques and applications*, IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 4, pp. 581–599, 2006.

[55] D. Joubert, W. Brink, *A mesh-based approach to incremental range image integration*, Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa, pp. 74–79, 2011.

[56] K. Kohara, N. Suganuma, *Obstacle detection based on occupancy grid maps from virtual disparity image*, ICCAS-SICE International Joint Conference, pp. 4617–4622, 2009.

[57] K. Konolige, *Improved occupancy grids for map building*, Autonomous Robots, vol. 4, pp. 351–367, 1997.

[58] D. Kortenkamp, T. Weymouth, *Topological mapping for mobile robots using a combination of sonar and vision sensing*, National Conference on Artificial Intelligence, pp. 979–984, 1994.

[59] G. Kraetzschmar, G. Gassull, K. Uhl, *Probabilistic quadtrees for variable-resolution mapping of large environments*, International Conference on Intelligent Autonomous Vehicles, pp. 653–658, 2004.

[60] B. Kuipers, Y. Byun, *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, Journal of Robotics and Autonomous Systems, vol. 8, no. 1, pp. 47–63, 1991.

[61] T. Lemaire, S. Lacroix, J. Solá, *A practical 3D bearing-only SLAM algorithm*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2449–2454, 2005.

[62] D. Lowe, *Object recognition from local scale-invariant features*, IEEE International Conference on Computer Vision, pp. 1150–1157, 1999.

[63] L. Matthies, A. Elfes, *Integration of sonar and stereo range data using a grid-based representation*, IEEE International Conference on Robotics and Automation, pp. 727–733, 1988.

[64] M. Maurette, *Mars rover autonomous navigation*, Autonomous Robots, vol. 14, pp. 199–208, 2003.

[65] N. Mitsou, C. Tzafestas, *Temporal occupancy grid for mobile robot dynamic environment mapping*, Mediterranean Conference on Control and Automation, pp. 1–8, 2007.

[66] J. Miura, Y. Negishi, Y. Shirai, *Mobile robot map generation by integrating omnidirectional stereo and laser range finder*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 250–255, 2002.

[67] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *FastSLAM: a factored solution to the simultaneous localization and mapping problem*, AAAI National Conference on Artificial Intelligence, pp. 593–598, 2002.

[68] H. Moravec, *Sensor fusion in certainty grids for mobile robots*, Artificial Intelligence Magazine, vol. 9, no. 2, pp. 61–74, 1988.

[69] H. Moravec, A. Elfes, *High resolution maps from wide angle sonar*, IEEE International Conference on Robotics and Automation, pp. 116–121, 1985.

[70] D. Murray, J. Little, *Using real-time stereo vision for mobile robot navigation*, Autonomous Robots, pp. 161–171, 2000.

[71] R. Murphy, J. Casper, M. Micire, J. Hyams, *Assessment of the NIST standard test bed for urban search and rescue*, PerMIS Workshop: Measuring the Performance and Intelligence of Systems, pp. 260–266, 2000.

[72] A. Nüchter, H. Surmann, S. Thrun, *6D SLAM with an application in autonomous mine mapping*, IEEE International Conference on Robotics and Automation, pp. 1998–2003, 2004.

[73] G. Oriolo, M. Vendittelli, G. Ulivi, *On-line map building and navigation for autonomous mobile robots*, IEEE International Conference on Robotics and Automation, pp. 2900–2906, 1995.

[74] D. Pagac, E. Nebot, H. Durrant-Whyte, *An evidential approach to map-building for autonomous vehicles*, IEEE International Conference on Robotics and Automation, pp. 623–629, 1998.

[75] K. Pathak, A. Birk, J. Poppinga, A. Schwertfeger, *3D forward sensor modeling and application to occupancy grid based sensor fusion*, IEEE International Conference of Intelligent Robots and Systems, pp. 2059–2064, 2007.

[76] P. Payeur, P. Hébert, D. Laurendeau, C. Gosselin, *Probabilistic octree modeling of a 3D dynamic environment*, IEEE International Conference on Robotics and Automation, pp. 1289–1296, 1997.

[77] M. Ribo, A. Pinz, *A comparison of three uncertainty calculi for building sonar-based occupancy grids*, Journal of Robotics and Autonomous Systems, vol. 35, no. 4, pp. 201–209, 2001.

[78] E. Richter, P. Lindner, G. Wanielik, *Advanced occupancy grid techniques for LIDAR based object detection and tracking*, IEEE Conference on Intelligent Transportation Systems, pp. 1–5, 2009.

[79] S. Ross, *A First Course in Probability Theory*, 7th edition, Pearson Education, 2006.

[80] R. Rubinstein, D. Kroese, *Simulation and the Monte Carlo Method*, John Wiley & Sons, 2008.

[81] J. Sasiadek, A. Monjazeb, D. Necsulescu, *Navigation of an autonomous mobile robot using EKF-SLAM and FastSLAM*, Mediterranean Conference on Control and Automation, pp. 517–522, 2008.

[82] B. Schiele, J. Crowley, *A comparison of position estimation techniques using occupancy grids*, IEEE International Conference on Robotics and Automation, pp. 1628–1634, 1994.

[83] P. Smith, I. Reid, A. Davison, *Real-time monocular SLAM with straight lines*, British Machine Vision Conference, pp. 17–26, 2006.

[84] J. Solá, A. Monin, M. Devy, T. Lemaire, *Undelayed initialization in bearing only SLAM*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2499–2504, 2005.

[85] M. Spiegel, J. Liu, *Mathematical Handbook of Formulas and Tables*, 2nd edition, McGraw-Hill, 1999.

[86] C. Stachniss, D. Hahnel, W. Burgard, *Exploration with active loop-closing for FastSLAM*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1505–1510, 2004.

[87] H. Surmann, K. Lingemann, A. Nüchter, J. Hertzberg, *A 3D laser range finder for autonomous mobile robots*, International Symposium on Robotics, pp. 153–158, 2001.

[88] H. Surmann, K. Lingemann, A. Nüchter, J. Hertzberg, *Fast acquiring and analysis of three dimensional laser range data*, International Workshop on Vision, Modeling and Visualization, pp. 59–66, 2001.

[89] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, 2008.

[90] S. Thrun, *Exploration and model building in mobile robot domains*, IEEE International Conference on Neural Networks, pp. 175–180, 1993.

[91] S. Thrun, *Learning metric-topological maps for indoor mobile robot navigation*, Artificial Intelligence, vol. 99, no. 1, pp. 21–71, 1998.

[92] S. Thrun, *Learning occupancy grids with forward models*, IEEE International Conference of Intelligent Robots and Systems, pp. 1676–1681, 2001.

[93] S. Thrun, *Robotic mapping: a survey*, Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nebel (editors), Morgan Kaufmann, 2002.

[94] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz, *Probabilistic algorithms and the interactive museum tour-guide robot minerva*, International Journal of Robotics Research, vol. 19, no. 11, pp. 972–999, 2000.

[95] S. Thrun, W. Burgard, D. Fox, *A probabilistic appraoch to concurrent mapping and localization for mobile robots*, Autonomous Robots, vol. 5, pp. 253–271, 1998.

[96] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[97] S. Thrun, J. Gutmann, D. Fox, W. Burgard, B. Kuipers, *Integrating topological and metric maps for mobile robot navigation: a statistical approach*, AAAI National Conference on Artificial Intelligence, pp. 989–995, 1998.

[98] L. Torabi, M. Kazemi, K. Gupta, *Configuration space based efficient view planning and exploration with occupancy grids*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2827–2832, 2007.

[99] G. Turk, M. Levoy, *Zippered polygon meshes from range images*, Proceedings of SIGGRAPH, pp. 311–318, 1994.

[100] M. Ugarte, A. Militino, A. Arnholt, *Probability and Statistics with R*, CRC Press, 2008.

[101] J. Wang, M. Oliveira, *A hole-filling strategy for reconstruction of smooth surfaces in range images*, Brazilian Symposium on Computer Graphics and Image Processing, pp. 11–18, 2003.

[102] L. Wasserman, *All of Statistics, a Concise Course in Statistical Inference*, 2nd edition, Springer Science and Business Media, 2005.

[103] Z. Wood, P. Schröder, D. Breen, M. Desbrun, *Semi-regular mesh extraction from volumes*, IEEE Visualization, pp. 275–282, 2000.

[104] K. Wurm, D. Hennes, D. Holz, R. Rusu, C. Stachniss, K. Konolige, W. Burgard, *Hierarchies of octrees for efficient 3D mapping*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4249–4255, 2011.

[105] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, W. Burgard, *Octomap: a probabilistic, flexible and compact 3D map representation for robotic systems*, IEEE ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010.

[106] B. Yamauchi, *A frontier-based approach for autonomous exploration*, IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 146–151, 1997.

[107] B. Yamauchi, P. Langley, *Place recognition in dynamic environments*, Journal of Robotic Systems, vol. 14, no. 2, pp. 107–120, 1997.

[108] M. Yguel, O. Aycard, C. Laugier, *Update policy of dense maps: efficient algorithms and sparse representation*, International Conference in Field and Service Robotics, pp. 23–33, 2007.

[109] R. Zask, M. Dailey, *Rapid 3D visualization of indoor scenes using 3D occupancy grid isosurfaces*, International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp. 672–675, 2009.